Finite elements with neural networks for the inverse elastography problem

Rafael Henriques[1*] and Sílvia Barbeiro[1]

[1]University of Coimbra, Department of Mathematics, CMUC, 3000-143, Coimbra, Portugal.

*Corresponding author(s). E-mail(s): rafael.henriques@mat.uc.pt, https://orcid.org/0000-0003-4173-8469;

Contributing authors: silvia@mat.uc.pt, https://orcid.org/0000-0002-2651-5083;

Abstract

In this work we investigate a mathematical model to reconstruct the mechanical properties of an heterogeneous elastic medium for the optical coherence elastography imaging modality. To this end, we propose machine learning tools by exploring neural networks to solve the inverse problem of elastography. Our algorithm updates the parameters combining the backpropagation technique with the ADAM optimizer to minimize a cost function which is defined using a fully discretized scheme of the direct problem. In our framework, we analyze the theoretical relative error between the exact solution and the numerical approximation given by the respective algorithm for the case of noise free data and noisy data. We report several computational results using fabricated data with and without noise.

Keywords: Finite element method, inverse problem, linear elasticity, mechanical properties reconstruction, neural networks, optical coherence elastography

1 Introduction

Optical coherence elastography (OCE) is a non-invasive emerging imaging technique that uses low-coherence interferometry to measure the mechanical properties of tissues, such as their elasticity and viscosity. It is based in optical coherence tomography (OCT) imaging modality to form pictures of biological tissue and map its biomechanical properties. OCE combines mechanical excitation with OCT for measuring the corresponding elastic displacement [6, 15, 19, 29]. An acoustic excitation system can be used for inducing the mechanical load to the medium where an ultrasound source is coupled with an OCT device. OCE has potential applications in many areas of medicine, including cancer diagnosis and treatment monitoring, by providing important information about the health of the tissue [27, 28].

Classical formulations aim to reconstruct the mechanical properties of the medium by solving the inverse problem of elastography. These approaches typically involve minimizing the difference between the measured and simulated displacements (*e.g.* [3, 7, 18, 26]), which besides being computational expensive, are not robust when the measurements contain noise. To overcome this limitations, we use neural networks based in the information of the direct problem. The neural network is designed to characterize the spatial mechanical properties of the medium and the idea is to minimize the objective function using the ADAM algorithm such that the given data solves the direct problem [9, 13, 21, 24]. The advantage is that neural networks, by training on large data sets, can learn to accurately predict the mechanical properties in a more efficient way, even in cases where the available data do not match exactly the physics of the system (e.g. the case of noisy data) [30].

Instead to what is usually found in literature [13, 21], our aim was not the creation of a black-box algorithm but to describe the mathematical foundations of the proposed method. In fact we present all the steps in detail including the deduction of the derivatives of the objective function with respect to the parameters we want to minimize. Another step forward not included in the mentioned papers concern the generalization of the parameters to space dependent functions which is also our objective in this work.

In both, the direct and the inverse problems, we consider the medium as a material with linear isotropic mechanical behavior, purely elastic.

We will explore the displacement field defined on piecewise linear function spaces, a commonly chosen approach for basis functions in the context of the finite element method (FEM). In situations involving nearly incompressible materials, *i.e.* with Poisson's ratio ν close to 0.5, the classical FEM scheme's performance may degrade due to locking as $\nu \to 0.5$ [1]. Here we are assuming that we are dealing with media for which the range of values of the Poisson's ratio leads to locking-free FEM solutions. A worthy application in this context is the aortic elastography [10]. For materials where locking is a challenge, various numerical methods have been proposed in the literature, including some variations of mixed finite element methods [5].

The article is organized as follows. In Section 2 we describe the mathematical model for the direct problem. We consider heterogeneous media and the model is based on the time-dependent linear elastic equation. We compute the corresponding numerical solution using a fully explicit in time finite element method in a three-dimensional space domain. In Section 3 we investigate the inverse problem through which we intend to infer the mechanical properties of the medium knowing the mechanical deformations by using neural networks. Here, we start by describing the optimization problem that we need to solve. Then we present the structure of the neural network and we describe the algorithm to solve the resulting optimization program. We also study the relative error between the exact solution and the approximation given by the neural network. Finally, in Section 4, we present several computational results for the inverse problem using fabricated data. We include experiments with noise free data and noisy data.

2 Elasticity model

In this chapter we describe the direct problem. For the formulation we use the time-dependent linear elasticity equation since it provides the connection between the mechanical properties and displacements induced by a mechanical excitation.

We start by introducing some notation needed. Let p be a scalar function, $\mathbf{v} = (v_i)_{1 \le i \le 3}$ a vector function and $\mathbf{A} = (a_{ij})_{1 \le i,j \le 3}$ a matrix of functions of three variables, all defined in a bounded domain $\Omega \subseteq \mathbb{R}^3$. We make use of the following Lebesgue spaces, for scalar, vector and matrix of functions given respectively by

$$L^2(\Omega) = \left\{p : ||p||_{L^2(\Omega)} < \infty\right\}, \quad L^2(\Omega) = \left\{\mathbf{v} : ||\mathbf{v}||_{L^2(\Omega)} < \infty\right\}$$

and

$$L^2(\Omega) = \left\{\mathbf{A} : ||\mathbf{A}||_{L^2(\Omega)} < \infty\right\},$$

where the corresponding norms are given by

$$||p||_{L^2(\Omega)} = (p,p)^{1/2}_{L^2(\Omega)}, \quad \|\mathbf{v}\|_{L^2(\Omega)} = (\mathbf{v},\mathbf{v})^{1/2}_{L^2(\Omega)} \text{ and } \|\mathbf{A}\|_{L^2(\Omega)} = (\mathbf{A}:\mathbf{A})^{1/2}_{L^2(\Omega)}.$$

The spaces $L^2(\Omega)$ are endowed with the inner products

$$(p,q)_{L^2(\Omega)} = \int_\Omega p\, q\, dx, \qquad (\mathbf{u}\,,\mathbf{v})_{L^2(\Omega)} = \int_\Omega \mathbf{u}\cdot\mathbf{v}\, dx = \sum_{i=1}^{3}\int_\Omega u_i v_i\, dx$$

and

$$(\mathbf{A}:\mathbf{B})_{L^2(\Omega)} = \int_\Omega \mathbf{A}:\mathbf{B}\, dx = \sum_{1\le i,j\le 3}\int_\Omega a_{ij}b_{ij}\, dx,$$

which induce the norms defined before. The space

$$H^1(\Omega) = \left\{\mathbf{u} : \mathbf{u} \in L^2(\Omega) \wedge \nabla\mathbf{u} \in L^2(\Omega)\right\}$$

is equipped with the following inner product

$$(\mathbf{u},\mathbf{v})_{H^1(\Omega)} = (\nabla\mathbf{u} : \nabla\mathbf{v})_{L^2(\Omega)} + (\mathbf{u},\mathbf{v})_{L^2(\Omega)}.$$

2.1 Time-dependent linear elasticity equation

Let us consider a heterogeneous isotropic elastic material which occupies the region $\overline{\Omega} \subseteq \mathbb{R}^3$, being Ω a polyhedron with boundary $\partial\Omega$. Let $\mathbf{u}(x,t)$ be the displacement field with $x = (x_1, x_2, x_3) \in \Omega$ and $t \in \mathbb{R}_0^+$. So the time-dependent elastic deformation $\mathbf{u}(x,t)$ when the medium properties are space dependent is given by

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2}(x,t) = \nabla \cdot \sigma(\mathbf{u}(x,t)\,; x) + \mathbf{f}(x,t), \quad (x,t) \in \Omega \times \mathbb{R}^+ \tag{1}$$

where

$$\sigma(\mathbf{u}(x,t)\,; x) = 2\mu(x)\varepsilon(\mathbf{u}(x,t)) + \lambda(x)\mathrm{tr}(\varepsilon(\mathbf{u}(x,t)))\mathrm{I}_3,$$

is the stress tensor defined with space dependent Lamé functions [25], ρ is the material density and $\mathbf{f}(x,t)$ is the force that acts on the volume of the body Ω [8]. Here ε is the strain tensor

$$\varepsilon(\mathbf{u}) = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^{\intercal}\right),$$

$\mathrm{tr}(\varepsilon(\mathbf{u}))$ is the trace of $\varepsilon(\mathbf{u})$ and I is the 3×3 identity matrix. In this work, we consider that functions $\mu(x)$ and $\lambda(x)$ are given, respectively, by

$$\mu(x) = \frac{E(x)}{2\,(1+\upsilon(x))},\ \lambda(x) = \frac{\upsilon(x)E(x)}{(1+\upsilon(x))\,(1-2\upsilon(x))}, \tag{2}$$

being $E(x)$ and $\upsilon(x)$ the functions representing the Young's Modulus and the Poisson's ratio respectively. We also assume that $\mu(x) \in [\mu_1, \mu_2]$ for all $x \in \Omega$, where $0 < \mu_1 < \mu_2$ and $\lambda(x) \in (0, \infty)$ for all $x \in \Omega$.

For the boundary of the set Ω, let us consider the partition $\partial\Omega = \Gamma_1 \cup \Gamma_2$ where $\Gamma_1 \cap \Gamma_2 = \emptyset$ and $\mathrm{meas}(\Gamma_2) > 0$. We impose the traction boundary condition on Γ_1,

$$\sigma(\mathbf{u}(x,t))\mathbf{n} = \mathbf{g}(x,t) \ \text{ on } \Gamma_1 \times \mathbb{R}^+, \tag{3}$$

being $\mathbf{n}$ the unit outer normal and $\mathbf{g}$ a vector function which represents the force applied on Γ_1. We also impose the null displacement boundary condition on Γ_2, that is,

$$\mathbf{u}(x,t) = 0 \text{ in } \Gamma_2 \times \mathbb{R}^+. \tag{4}$$

The model (1), (3), (4) is completed with the initial condition

$$\mathbf{u}(x,0) = 0 \text{ in } \overline{\Omega}. \tag{5}$$

2.2 Numerical solution

Now we will discuss the numerical approximation of the above problem. Let us consider a partition of Ω into M tetrahedra K_j, $j \in \{1, ..., M\}$ such that

$$\overline{\Omega} = \bigcup_{j=1}^{M} K_j. \tag{6}$$

The resulting partition is denoted by Ω_h where h represents its diameter. For any pair of tetrahedra in the partition either they don't intersect or they have in common only vertices or edges.

For $V = \left\{\mathbf{v} \in H^1(\Omega) : \mathbf{v}|_{\Gamma_2} = 0\right\}$, let us consider the finite dimensional subspace $V_h \subset V$ of continuous piecewise linear functions on each tetrahedron. Assuming that N is the total number of vertices associated with the tetrahedra in Ω_h then $\dim V_h = 3N$. The semi-discrete finite element formulation of the problem (1), (3), (4) and (5) consist of finding $\mathbf{u}_h(t) \in V_h$ such that

$$\begin{cases} \rho \dfrac{\mathrm{d}^2}{\mathrm{d}t^2}(\mathbf{u}_h(t), \mathbf{v}_h)_{L^2(\Omega)} + a(\mathbf{u}_h(t), \mathbf{v}_h) &= l(\mathbf{v}_h), \;\; \forall \mathbf{v}_h \in V_h,\, t \in \mathbb{R}^+, \\ \mathbf{u}_h(0) &= 0 \text{ in } \overline{\Omega}, \end{cases} \tag{7}$$

where

$$a(\mathbf{u}_h(t), \mathbf{v}_h) = \int_\Omega 2\mu(x)\varepsilon(\mathbf{u}_h(t)) : \varepsilon(\mathbf{v}_h) + \lambda(x)(\nabla \cdot \mathbf{u}_h(t))(\nabla \cdot \mathbf{v}_h)\, dx,$$

and

$$l(\mathbf{v}) = (\mathbf{g}(t), \mathbf{v})_{L^2(\Gamma_1)} + (\mathbf{f}(t), \mathbf{v})_{L^2(\Omega)}. \tag{8}$$

The inner products in (8) represent the following integrals

$$(\mathbf{g}(t), \mathbf{v})_{L^2(\Gamma_1)} = \int_{\Gamma_1} \mathbf{g}(t) \cdot \mathbf{v}\, ds$$

and

$$(\mathbf{f}(t), \mathbf{v})_{L^2(\Omega)} = \int_\Omega \mathbf{f}(t) \cdot \mathbf{v}\, dx,$$

respectively. Here we are assuming that $\mathbf{g} \in L^2(\Gamma_1)$ and $\mathbf{f} \in H^{-1}(\Omega)$. For results about uniqueness and existence of this discrete problem see for example [22].

Let us associate to each vertex x^s of the partition Ω_h in tetrahedra three base functions ϕ_{sr}, $s \in \{1, ..., N\}$, $r \in \{1, 2, 3\}$. These functions are continuous in $\overline{\Omega}$ and linear in each tetrahedron, such that $\phi_{sr}(x^s) = 1$, $\phi_{sr}(x^k) = 0$ $(k \neq s)$ and the support of ϕ_{sr} consists in all tetrahedra that share x^s as a vertex. We have

$$V_h = \operatorname{span}\{\phi_{11}, ..., \phi_{N1}, \phi_{12}, ..., \phi_{N2}, \phi_{13}, ..., \phi_{N3}\}.$$

In this way, each component of the approximate solution $\mathbf{u}_h = (u_{h1}, u_{h2}, u_{h3}) \in V_h$ can be written as a linear combination of the basis functions ϕ_{sr} with

$$u_{hr}(x,y,z,t) = \sum_{s=1}^{N} U_{sr}(t)\phi_{sr}(x,y,z),\ r \in \{1,2,3\}\,.$$

For the discretization in time, we denote by $\mathbf{u}_h^n$ the value of the variable $\mathbf{u}_h$ at time $t^n = n\Delta t$, $n = 0,1,2,\cdots$ with $\Delta t > 0$. So each component of the approximate solution $\mathbf{u}_h^n = (u_{h1}^n, u_{h2}^n, u_{h3}^n)$ will be written as

$$u_{hr}^n(x,y,z,t^n) = \sum_{s=1}^{N} U_{sr}^n \phi_{sr}\,(x,y,z)\,,\ r \in \{1,2,3\}\,,$$

being U_{sr}^n, $r \in \{1,2,3\}$, $s \in \{1,...,N\}$, the coefficients that we want to compute. Using an explicit scheme in time for the semi-discrete model (7), we obtain the following fully discretized scheme on space and time: find $\mathbf{u}_h^n \in V_h$, $n = 0,1,2,\cdots$, such that

$$\begin{cases} \dfrac{\rho}{\Delta t^2}(\mathbf{u}_h^{n+1} - 2\mathbf{u}_h^n + \mathbf{u}_h^{n-1}, \mathbf{v}_h)_{L^2(\Omega)} + a(\mathbf{u}_h^n, \mathbf{v}_h) &= (\mathbf{g}^n, \mathbf{v}_h)_{L^2(\Gamma_1)} \\ & \quad +(\mathbf{f}^n, \mathbf{v}_h)_{L^2(\Omega)}, \\ \mathbf{u}_h^0 &= 0 \text{ in } \overline{\Omega}, \end{cases} \tag{9}$$

for all $\mathbf{v}_h \in V_h$. We can write the problem (9) with respect boundary conditions, in matrix form, as follows: find U^n [2]

$$U^n = [U_{11}^n, ..., U_{N1}^n, U_{12}^n, ..., U_{N2}^n, U_{13}^n, ..., U_{N3}^n]^\intercal$$

such that

$$\begin{aligned} \frac{\rho}{\Delta t^2} B(U^{n+1} - 2U^n + U^{n-1}) + A_{\mu,\lambda} U^n &= G^n + F^n, n = 0,1,2,\cdots, \\ U^{-1} = U^0 &= 0, \end{aligned} \tag{10}$$

where B is the $3N \times 3N$ mass matrix and $A_{\mu,\lambda}$ is the well-known stiffness matrix of size $3N \times 3N$ [22]. Approximating $\mu(x)$ and $\lambda(x)$ by the corresponding values in the mid point $\mathbf{x}_p = (x_{1p}, x_{2p}, x_{3p})^\intercal$, $p \in \{1,...,M\}$ of each tetrahedron, that is, $\mu(\mathbf{x}_p)$ and $\lambda(\mathbf{x}_p)$ respectively, the matrix $A_{\mu,\lambda}$ in (10) can be written as (see more details in [2])

$$A_{\mu,\lambda} = \sum_{p=1}^{M} A_\mu^{K_p}\mu(\mathbf{x}_p) + A_\lambda^{K_p}\lambda(\mathbf{x}_p). \tag{11}$$

Let $A_\mu^K = \left[A_\mu^{K_1}, A_\mu^{K_2}, \cdots, A_\mu^{K_M}\right]$ and $A_\lambda^K = \left[A_\lambda^{K_1}, A_\lambda^{K_2}, \cdots, A_\lambda^{K_M}\right]$ be matrices with size $3N \times (3N \times M)$ where $A_\mu^{K_p}$ and $A_\lambda^{K_p}$ are the $3N \times 3N$ matrices in (11). Let

$\mathbf{U}^n$ a matrix with size $(3N \times M) \times M$, where in the first column U^n occupies the first $3N$ entries, in the second column U^n occupies the entries from $3N+1$ until $3N \times 2$, and so on. In addition let $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ be $M \times 1$ vectors with entries $\mu(\mathbf{x}_p)$ and $\lambda(\mathbf{x}_p)$, $p \in \{1, \cdots, M\}$ respectively. In this way, we can write

$$\begin{aligned} A_{\mu,\lambda} U^n &= A_\mu^K \mathbf{U}^n \boldsymbol{\mu} + A_\lambda^K \mathbf{U}^n \boldsymbol{\lambda} \\ &= A^n \left[\boldsymbol{\mu} \ \boldsymbol{\lambda}\right]^\intercal \end{aligned}$$

where

$$A^n = \left[A_\mu^K \mathbf{U}^n \ \ A_\lambda^K \mathbf{U}^n\right] \tag{12}$$

is a matrix with size $3N \times 2M$.

3 Inverse Problem

In this section, we formulate the inverse problem as an optimization problem to determine neural networks. The objective function uses the information of the scheme (10) and the goal is to infer the approximations $\mu_{\boldsymbol{\theta}}(x)$, $\lambda_{\boldsymbol{\theta}}(x)$ of the exact solution $\mu(x)$, $\lambda(x)$ so that the objective function is minimized. The subscript $\boldsymbol{\theta}$ refers to the set of all parameters of the neural network involved in our formulation which will be detailed in what follows.

This section is divided in three subsections where we start by presenting the optimization problem, in Section 3.1. Then, in Section 3.2, we derive the derivatives of the objective function with respect to the parameters of the neural network and we describe the algorithm to update these parameters. Several results that establish bounds on the relative error for both noise free and noisy data are presented in Section 3.3.

3.1 Description of problem

We will consider the approximations $\mu_{\boldsymbol{\theta}}(x)$, $\lambda_{\boldsymbol{\theta}}(x)$ being neural networks with parameters θ, though our formulation also holds for a more general approximation (also e.g. chebyshev polynomials [17] or radial basis functions [23]).

For the theoretical result and real application we make use of diversified data. Therefore, we will consider that we have access to $Q(T+2)$ data sets either from simulation or experimental data corresponding to deformations, body force densities and applied traction of the form

$$\mathbf{u}_{ih}^n, \mathbf{f}_i^n, \mathbf{g}_i^n,\ i \in \{1, ..., Q\},\ n \in \{0, ..., T+1\}$$

where

$$\mathbf{u}_{ih}^n = \left(u_{ih1}^n, u_{ih2}^n, u_{ih3}^n\right),\ i \in \{1, ..., Q\},\ n \in \{0, ..., T+1\}.$$

The index i and n enumerate the data in space and time respectively. For each $i = 1, \cdots, Q$ and $n = 0, \cdots, T+1$ we write each function $u^n_{ih1}, u^n_{ih2}, u^n_{ih3}$ in the form

$$u^n_{ihr}(x, y, z, t^n) = \sum_{s=1}^{N} U^n_{isr} \phi_{sr}(x, y, z), \; r \in \{1, 2, 3\}. \tag{13}$$

Let $\boldsymbol{\mu_\theta}$ and $\boldsymbol{\lambda_\theta}$ be $M \times 1$ vectors with entries $\mu_{\boldsymbol{\theta}}(\mathbf{x}_p)$ and $\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p)$, $p \in \{1, ..., M\}$ respectively,

$$A^n_i = \left[A^K_\mu \mathbf{U}^n_i \;\; A^K_\lambda \mathbf{U}^n_i\right] \tag{14}$$

a matrix and T^n_i the vector given by

$$T^n_i = G^n_i + F^n_i - \frac{\rho}{\Delta t^2} B(U^{n+1}_i - 2U^n_i + U^{n-1}_i). \tag{15}$$

Then the inverse problem can be described by the following minimization problem:

$$\min_{\boldsymbol{\theta}} \sum_{n=1}^{T} \sum_{i=1}^{Q} \left\| A^n_i \left[\boldsymbol{\mu_\theta} \;\; \boldsymbol{\lambda_\theta}\right]^\intercal - T^n_i \right\|^2_{L^2_h(\Omega)}. \tag{16}$$

Here we use the discrete L^2_h-norm, defined for any $3N \times 1$ vector y, as

$$\|y\|^2_{L^2_h(\Omega)} = \sum_{K \in \Omega_h} \|y\|^2_{L^2_h(K)},$$

with

$$\|y\|^2_{L^2_h(K)} = \frac{|K|}{4} \sum_{i=1}^{4} \sum_{j=0}^{2} y^2_{t(r^K_i)+jN},$$

where $|K|$ denotes the volume of the tetrahedron K with vertices $r^K_i, i \in \{1, \ldots, 4\}$. The function t is defined by

$$\begin{array}{rcl} t : \mathbb{R}^3 & \rightarrow & \{1, ..., N\} \\ r^K_i & \mapsto & t(r^K_i), \end{array}$$

where $t(r^K_i)$ is the index that corresponds to vertex of r^K_i in the global numbering.

In what follows, we denote by $\boldsymbol{\theta}$ the set of all parameters of the neural networks that will be constructed and this set incorporates the weights $\mathbf{W}$ and bias $\mathbf{b}$, so $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$ [12]. We denote the objective function by $L_{\boldsymbol{\theta}}(\mathbf{W},\mathbf{b})$, that is, $L_{\boldsymbol{\theta}}(\mathbf{W},\mathbf{b}) = \sum_{n=1}^{T} \sum_{i=1}^{Q} \left\| A^n_i \left[\boldsymbol{\mu_\theta} \;\; \boldsymbol{\lambda_\theta}\right]^\intercal - T^n_i \right\|^2_{L^2_h(\Omega)}$ and along the text we will use the notation $L_{\boldsymbol{\theta}}$ or $L_{\boldsymbol{\theta}}(\mathbf{W},\mathbf{b})$ depending on which one is more convenient.

Let $\mathbb{I}$ be a $1 \times M$ vector with all entries equal to one, $\mathrm{diag}(\frac{|K|}{4})$ the diagonal matrix of size $M \times M$ with entries $\frac{|K_j|}{4}$, $j \in \{1, \cdots, M\}$ for some numbering of the tetrahedra, $\mathrm{diag}(A_i^n \left[\boldsymbol{\mu_\theta}\ \boldsymbol{\lambda_\theta}\right]^\intercal - T_i^n)$ the diagonal matrix of size $3N \times 3N$ with entries $(A_i^n \left[\boldsymbol{\mu_\theta}\ \boldsymbol{\lambda_\theta}\right]^\intercal - T_i^n)_{t(r_i^K)+jN}$ and $\boldsymbol{\Lambda} = [\Lambda, \Lambda, \Lambda]$ a $M \times 3N$ matrix where Λ is a $M \times N$ matrix with entries $\delta_{ji} = 1$ if and only if the vertex $r_i^K \in K_j$ and $\delta_{ji} = 0$ otherwise.

In this way, the objective function in (16) is given by

$$
\begin{aligned}
L_{\boldsymbol{\theta}}\left(\mathbf{W},\mathbf{b}\right) = & \sum_{n=1}^{T} \sum_{i=1}^{Q} \mathbb{I}\, \mathrm{diag}(\frac{|K|}{4})\ \boldsymbol{\Lambda}\ \mathrm{diag}(A_i^n \left[\boldsymbol{\mu_\theta}\ \boldsymbol{\lambda_\theta}\right]^\intercal - T_i^n) \times \\
& \times \left(A_i^n \left[\boldsymbol{\mu_\theta}\ \boldsymbol{\lambda_\theta}\right]^\intercal - T_i^n\right). \qquad (17)
\end{aligned}
$$

Our main goal is determine neural networks with parameters $\boldsymbol{\theta}$ such that the objective function $L_{\boldsymbol{\theta}}$ is minimized. In Section 3.2 we discuss an iterative optimization process and how to update these parameters to obtain the minimizer of (17).

By the universal approximation theorem [14], μ and λ can be approximated with arbitrary accuracy, by some neural networks $\mu_{\boldsymbol{\theta}}$ and $\lambda_{\boldsymbol{\theta}}$ respectively. In other words, this theorem is saying that for any tol > 0 there exist a neural network $\mu_{\boldsymbol{\theta}}$ and a neural network $\lambda_{\boldsymbol{\theta}}$ such that $\left\| \left[\boldsymbol{\mu_\theta} - \boldsymbol{\mu}\ \boldsymbol{\lambda_\theta} - \boldsymbol{\lambda}\right]^\intercal \right\|_\infty < \mathrm{tol}$.

3.2 Optimization method

In this section we compute the derivatives of the objective function and present the algorithm to solve the optimization problem (16).

To compute the values $\mu_{\boldsymbol{\theta}}(\mathbf{x}_p)$ and $\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p)$ in (17) we use feed-forward networks through the following transformation:

$$
\begin{aligned}
\mathbf{a}_p^0 &= \mathbf{c}_p^0 = \mathbf{x}_p, \\
\mathbf{a}_p^l &= \sigma^l\left(\mathbf{W}^l \mathbf{a}_p^{l-1} + \mathbf{b}^l\right), \; l \in \{1, ..., L_1 - 1\}, \\
\mathbf{c}_p^l &= \sigma^l\left(\mathbf{W}^l \mathbf{c}_p^{l-1} + \mathbf{b}^l\right), \; l \in \{1, ..., L_2 - 1\}, \\
\mu_{\boldsymbol{\theta}}(\mathbf{x}_p) &= \sigma^{L_1}\left(\mathbf{W}^{L_1} \mathbf{a}_p^{L_1-1} + \mathbf{b}^{L_1}\right), \\
\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p) &= \sigma^{L_2}\left(\mathbf{W}^{L_2} \mathbf{c}_p^{L_2-1} + \mathbf{b}^{L_2}\right), \qquad (18)
\end{aligned}
$$

where L_1 and L_2 are the number of layers of the neural network $\mu_{\boldsymbol{\theta}}$ and $\lambda_{\boldsymbol{\theta}}$ respectively. In our problem, the inputs are the points $\mathbf{x}_p$ and the targets are given by the values T_i^n.

In what follows we denote by L the number of layers which can be L_1 or L_2 depending if we are considering the neural network for $\mu_{\boldsymbol{\theta}}$ or $\lambda_{\boldsymbol{\theta}}$ respectively. In (18) $\mathbf{a}_p^l =$

$\left(a^l_{1p}, a^l_{2p}, \cdots, a^l_{m^{(l)}p}\right)^\intercal$, $\mathbf{b}^l = \left(b^l_1, b^l_2, \cdots, b^l_{m^{(l)}}\right)^\intercal$ and $\mathbf{c}^l_p = \left(c^l_{1p}, c^l_{2p}, \cdots, c^l_{m^{(l)}p}\right)^\intercal$ are vectors of dimension $m^{(l)} \times 1$ for $l \in \{1, ..., L-1\}$. $\mathbf{W}^l = \left(w^l_{ij}\right)$ is a matrix of dimension $m^{(l)} \times m^{(l-1)}$, $l \in \{1, ..., L\}$ where $m^{(l)}$, $l \in \{2, ..., L-1\}$ can be the size we want. However, the values $m^{(0)}$, $m^{(L)}$ are three and one, respectively, since $\mathbf{x}_p$ is a vector of dimension three and the network output value is a real number. Here $\mathbf{a}^0_p$ and $\mathbf{c}^0_p$ are vectors of dimension 3×1 and $\mathbf{b}^L$, $\mu_{\boldsymbol{\theta}}(\mathbf{x}_p)$ and $\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p)$ are real numbers. In this work we will consider the first $L-1$ hidden layers with non-linear activation functions σ^l, $l \in \{1, ..., L-1\}$ while for the output layer we will impose the linear activation function $\sigma^L(x) = x$.

For each $\mathbf{x}_p$, the feed-forward neural network in (18) propagates this information to the hidden units at each layer and produces the outputs $\mu_{\boldsymbol{\theta}}(\mathbf{x}_p)$ and $\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p)$ as it is shown in Figure 1. Therein we display a scheme for the forward propagation where information flows forward through the network. After this steps, we can evaluate the value of the objective function in (17). Therefore, our goal is to find a procedure for evaluating the derivatives of the error function with respect to the weights and biases in the network. To this end, we will use the so called backpropagation technique since it allows to propagate the error backwards through the network [11]. A valuable contribution of this technique is in providing a computationally efficient method for evaluating such derivatives [4].

To present the derivatives of the loss function we will simplify the notation in (18). Defining $\mathbf{z}^l_p = \mathbf{W}^l \mathbf{a}^{l-1}_p + \mathbf{b}^l$, $l \in \{1, ..., L\}$, $\mathbf{z}^l_p = (z^l_{ip})$, we can write $\mathbf{a}^l_p$ in (18) in the form

$$\mathbf{a}^l_p = \sigma^l\left(\mathbf{z}^l_p\right), \; l \in \{1, ..., L-1\} \tag{19}$$

$$\mathbf{a}^L_p = \mathbf{z}^l_p. \tag{20}$$

Considering $P^n_{\boldsymbol{\theta} i} = A^n_i \left[\boldsymbol{\mu_\theta} \; \boldsymbol{\lambda_\theta}\right]^\intercal$ observe that the objective function in (16) can be written by $L_{\boldsymbol{\theta}}\left(\mathbf{W}\text{,}\mathbf{b}\right) = \sum_{n=1}^{T}\sum_{i=1}^{Q} \hat{L}_{\boldsymbol{\theta}}(\mathbf{W}, \mathbf{b})$, where $\hat{L}_{\boldsymbol{\theta}}(\mathbf{W}, \mathbf{b})$ is the sum

$$\hat{L}_{\boldsymbol{\theta}}(\mathbf{W}, \mathbf{b}) = \sum_{m=1}^{M} \frac{|K_m|}{4} \sum_{l=1}^{4}\sum_{j=0}^{2} (P^n_{\boldsymbol{\theta} i} - T^n_i)^2_{t(r^{K_m}_l)+jN}.$$

We determine the derivatives for $\mu_{\boldsymbol{\theta}}$ since similar expressions can be obtained for $\lambda_{\boldsymbol{\theta}}$. To derive the derivatives we will start to analyze the terms in $\hat{L}_{\boldsymbol{\theta}}(\mathbf{W}, \mathbf{b})$ following the steps in [12]. In this way, we need to compute the terms:

$$\frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial w^l_{ij}} \text{ and } \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial b^l_i}.$$

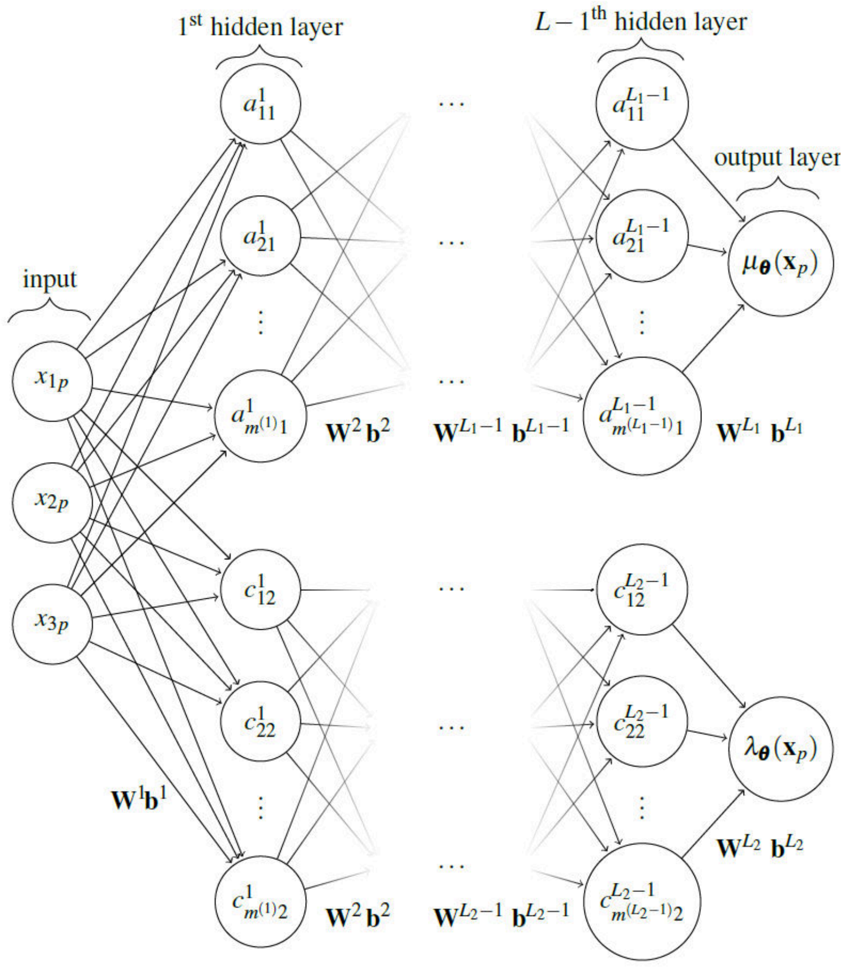

1st hidden layer
$L-1$th hidden layer
input
output layer
x_{1p}
x_{2p}
x_{3p}
a^1_{11}
a^1_{21}
$a^1_{m^{(1)}1}$
c^1_{12}
c^1_{22}
$c^1_{m^{(1)}2}$
$a^{L_1-1}_{11}$
$a^{L_1-1}_{21}$
$a^{L_1-1}_{m^{(L_1-1)}1}$
$c^{L_2-1}_{12}$
$c^{L_2-1}_{22}$
$c^{L_2-1}_{m^{(L_2-1)}2}$
$\mu_{\boldsymbol{\theta}}(\mathbf{x}_p)$
$\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p)$
$\mathbf{W}^1\mathbf{b}^1$
$\mathbf{W}^2\,\mathbf{b}^2$
$\mathbf{W}^{L_1-1}\,\mathbf{b}^{L_1-1}$
$\mathbf{W}^{L_1}\,\mathbf{b}^{L_1}$
$\mathbf{W}^{L_2-1}\,\mathbf{b}^{L_2-1}$
$\mathbf{W}^{L_2}\,\mathbf{b}^{L_2}$

Fig. 1: Scheme of the L_1-layer and L_2-layer networks, with inputs $\mathbf{x}_p$, to obtain the values $\mu_{\boldsymbol{\theta}}$ and $\lambda_{\boldsymbol{\theta}}$ respectively.

Using chain rule for partial derivatives we obtain

$$\frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial w^l_{ij}} = \sum_{p=1}^{M} \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial z^l_{ip}} \times \frac{\partial z^l_{ip}}{\partial w^l_{ij}} \tag{21}$$

and

$$\frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial b^l_{i}} = \sum_{p=1}^{M} \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial z^l_{ip}} \times \frac{\partial z^l_{ip}}{\partial b^l_{i}}. \tag{22}$$

Since z_{ip}^l is a linear function of the weights w_{ij}^l and bias b_i^l in that layer, we know that

$$z_{ip}^l = \sum_{j=1}^{m^{(l-1)}} w_{ij}^l a_{jp}^{l-1} + b_i^l,\, p \in \{1, \cdots, M\},$$

and then

$$\frac{\partial z_{ip}^l}{\partial w_{ij}^l} = a_{jp}^{l-1}, \frac{\partial z_{ip}^l}{\partial b_i^l} = 1.$$

If we define $s_{ip}^l \equiv \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial z_{ip}^l}$ being the sensitivity of $\hat{L}_{\boldsymbol{\theta}}$ to changes in the i-th element of $\mathbf{z}_p^l$, then the derivatives in (21) and (22) take the following form:

$$\frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial w_{ij}^l} = \sum_{p=1}^{M} s_{ip}^l a_{jp}^{l-1} \text{ and } \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial b_i^l} = \sum_{p=1}^{M} s_{ip}^l.$$

In matrix form this becomes:

$$\frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{W}^l} = \sum_{p=1}^{M} \mathbf{s}_p^l \left(\mathbf{a}_p^{l-1}\right)^{\intercal} \text{ and } \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{b}^l} = \sum_{p=1}^{M} \mathbf{s}_p^l,$$

where

$$\mathbf{s}_p^l \equiv \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{z}_p^l} = \begin{bmatrix} \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial z_{1p}^l} \\ \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial z_{2p}^l} \\ \vdots \\ \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial z_{m^{(l)}p}^l} \end{bmatrix}.$$

Now we will derive a recurrence relationship between $\mathbf{s}_p^l$ at layer l and the sensitivity $\mathbf{s}_p^{l+1}$ at layer $l+1$. Using chain rule in matrix form we have

$$\mathbf{s}_p^l = \left(\frac{\partial \mathbf{z}_p^{l+1}}{\partial \mathbf{z}_p^l}\right)^T \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{z}_p^{l+1}} = \left(\frac{\partial \mathbf{z}_p^{l+1}}{\partial \mathbf{z}_p^l}\right)^T \mathbf{s}_p^{l+1}. \tag{23}$$

The first term in (23) is the following Jacobian matrix:

$$\frac{\partial \mathbf{z}_p^{l+1}}{\partial \mathbf{z}_p^l} = \begin{bmatrix} \frac{\partial z_{1p}^{l+1}}{\partial z_{1p}^l} & \frac{\partial z_{1p}^{l+1}}{\partial z_{2p}^l} & \cdots & \frac{\partial z_{1p}^{l+1}}{\partial z_{m^{(l)}p}^l} \\ \frac{\partial z_{2p}^{l+1}}{\partial z_{1p}^l} & \frac{\partial z_{2p}^{l+1}}{\partial z_{2p}^l} & \cdots & \frac{\partial z_{2p}^{l+1}}{\partial z_{m^{(l)}p}^l} \\ \vdots & \vdots & & \vdots \\ \frac{\partial z_{m^{(l+1)}p}^{l+1}}{\partial z_{1p}^l} & \frac{\partial z_{m^{(l+1)}p}^{l+1}}{\partial z_{2p}^l} & \cdots & \frac{\partial z_{m^{(l+1)}p}^{l+1}}{\partial z_{m^{(l)}p}^l} \end{bmatrix}.$$

Note that the i, j element of the matrix $\frac{\partial \mathbf{z}_p^{l+1}}{\partial \mathbf{z}_p^l}$ is

$$\frac{\partial z_{ip}^{l+1}}{\partial z_{jp}^l} = \frac{\partial \left(\sum_{q=1}^{m^{(l)}} w_{iq}^{l+1} a_{qp}^l + b_i^{l+1}\right)}{\partial z_{jp}^l} = w_{ij}^{l+1} \frac{a_{jp}^l}{\partial z_{jp}^l} = w_{ij}^{l+1} \frac{\partial \sigma^l \left(z_{jp}^l\right)}{\partial z_{jp}^l}.$$

In this way, the Jacobian matrix can be written as

$$\frac{\partial \mathbf{z}_p^{l+1}}{\partial \mathbf{z}_p^l} = \mathbf{W}^{l+1} \mathrm{diag}\left(\frac{\partial \sigma^l \left(\mathbf{z}_p^l\right)}{\partial \mathbf{z}_p^l}\right),$$

being $\mathrm{diag}\left(\frac{\partial \sigma^l (\mathbf{z}_p^l)}{\partial \mathbf{z}_p^l}\right)$ a diagonal matrix of size $m^{(l)} \times m^{(l)}$ with diagonal elements $\frac{\partial \sigma^l (z_{jp}^l)}{\partial z_{jp}^l}$, $j \in \left\{1, \cdots, m^{(l)}\right\}$. Therefore (23) takes the form

$$\mathbf{s}_p^l = \mathrm{diag}\left(\frac{\partial \sigma^l \left(\mathbf{z}_p^l\right)}{\partial \mathbf{z}_p^l}\right) \left(\mathbf{W}^{l+1}\right)^\intercal \mathbf{s}_p^{l+1}, \text{for } l \in \{L-1, \cdots, 1\}. \tag{24}$$

The next step is to compute the sensitivity in last layer, that is, $\mathbf{s}_p^L$. First we will determined the term $\mathbf{s}_p^L$ with respect to $\mu_{\boldsymbol{\theta}}$. We have

$$\mathbf{s}_p^L = \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \left(P_{\boldsymbol{\theta} i}^n\right)_{t(r_l^{K_m})+jN}} \frac{\partial \left(P_{\boldsymbol{\theta} i}^n\right)_{t(r_l^{K_m})+jN}}{\partial \mathbf{a}_p^L} \frac{\partial \mathbf{a}_p^L}{\partial \mathbf{z}_p^L}. \tag{25}$$

Note that from (20) $\frac{\partial \mathbf{a}_p^L}{\partial \mathbf{z}_p^L} = 1$. Let $\left(\mu_{isp}^n\right)_{1 \leq s \leq 3N, 1 \leq p \leq M}$ and $\left(\lambda_{isp}^n\right)_{1 \leq s \leq 3N, 1 \leq p \leq M}$ be the entries of the matrices $A_\mu^K \mathbf{U}_i^n$ and $A_\lambda^K \mathbf{U}_i^n$ respectively. We write

$$\left(P_{\boldsymbol{\theta} i}^n\right)_{t(r_l^{K_m})+jN} = \sum_{p=1}^{M} \mu_{i\left(t(r_l^{K_m})+jN\right)p}^n \mathbf{a}_p^L + \lambda_{i\left(t(r_l^{K_m})+jN\right)p}^n \mathbf{c}_p^L,$$

and

$$\frac{\partial \left(P^n_{\boldsymbol{\theta} i}\right)_{t(r_l^{K_m})+jN}}{\partial \mathbf{a}^L_p} = \mu^n_{i\left(t(r_l^{K_m})+jN\right)p}.$$

Let e_p be a unitary vector with dimension $M \times 1$ where the component p is equal to one. Then $\mu^n_{i\left(t(r_l^{K_m})+jN\right)p} = \left(A^K_\mu \mathbf{U}^n_i e_p\right)_{t(r_l^{K_m})+jN}$ and (25) takes the form

$$\mathbf{s}^L_p = \mathbb{I}\ \mathrm{diag}(\frac{|K|}{2})\ \boldsymbol{\Lambda}\ \mathrm{diag}(P^n_{\boldsymbol{\theta} i} - T^n_i) A^K_\mu \mathbf{U}^n_i e_p.$$

We also can organize this information in the following vector

$$\mathbf{s}^L = \begin{bmatrix} \mathbf{s}^L_1 \\ \mathbf{s}^L_2 \\ \vdots \\ \mathbf{s}^L_M \end{bmatrix}$$

where the component p of $\mathbf{s}^L$ is given by $\left(\mathbf{s}^L\right)_p = \mathbf{s}^L_p$. In this way, we get

$$\mathbf{s}^L = \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{z}^L} = \left(\frac{\partial \mathbf{P}^n_{\boldsymbol{\theta} i}}{\partial \mathbf{z}^L}\right)^\intercal \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{P}^n_{\boldsymbol{\theta} i}}.$$

Then

$$\mathbf{s}^L = (\mathbf{U}^n_i)^\intercal \left(A^K_\mu\right)^\intercal \mathrm{diag}(P^n_{\boldsymbol{\theta} i} - T^n_i)\ \boldsymbol{\Lambda}^\intercal\ \mathrm{diag}(\frac{|K|}{2})\ \mathbb{I}^\intercal. \tag{26}$$

Finally the derivatives can be summarized as follows:

$$\begin{aligned} \mathbf{s}^L &= (\mathbf{U}^n_i)^\intercal \left(A^K_\mu\right)^\intercal \mathrm{diag}(P^n_{\boldsymbol{\theta} i} - T^n_i)\ \boldsymbol{\Lambda}^\intercal\ \mathrm{diag}(\frac{|K|}{2})\ \mathbb{I}^\intercal, \\ \mathbf{s}^{L-1}_p &= \mathrm{diag}\left(\frac{\partial \sigma^{L-1}\left(\mathbf{z}^{L-1}_p\right)}{\partial \mathbf{z}^{L-1}_p}\right) \left(\mathbf{W}^L\right)^\intercal \left(\mathbf{s}^L\right)_p, \\ \mathbf{s}^l_p &= \mathrm{diag}\left(\frac{\partial \sigma^l\left(\mathbf{z}^l_p\right)}{\partial \mathbf{z}^l_p}\right) \left(\mathbf{W}^{l+1}\right)^\intercal \mathbf{s}^{l+1}_p,\ \text{for } l \in \{L-2, \cdots, 1\}, \end{aligned} \tag{27}$$

with

$$\begin{aligned} \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{W}^L} &= \sum_{p=1}^{M} \left(\mathbf{s}^L\right)_p \left(\mathbf{a}^{L-1}_p\right)^\intercal, \\ \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{b}^L} &= \sum_{p=1}^{M} \left(\mathbf{s}^L\right)_p, \end{aligned}$$

$$
\begin{aligned}
\frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{W}^l} &= \sum_{p=1}^{M} \mathbf{s}_p^l \left(\mathbf{a}_p^{l-1}\right)^{\intercal}, \text{ for } l \in \{L-1, \cdots, 1\}, \\
\frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{b}^l} &= \sum_{p=1}^{M} \mathbf{s}_p^l, \text{ for } l \in \{L-1, \cdots, 1\}.
\end{aligned} \tag{28}
$$

In this way, the gradient vector $\nabla \hat{L}_{\boldsymbol{\theta}}$ is given by

$$
\nabla \hat{L}_{\boldsymbol{\theta}} = \left[\frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{W}^1}, \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{b}^1}, \dots, \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{W}^L}, \frac{\partial \hat{L}_{\boldsymbol{\theta}}}{\partial \mathbf{b}^L}\right]^{\intercal}. \tag{29}
$$

Note that the expressions for the derivatives in terms of $\lambda_{\boldsymbol{\theta}}$ can be derived in the same way by replacing A_μ^K in (26) by A_λ^K.

After obtaining the derivatives of the objective function with respect to the parameters in the network we are ready to present the overall algorithm which is an iterative process. For updating the parameters, there are several optimization methods available in the literature as the conjugate gradient algorithm, the Newton's method, the Levenberg-Marquardt algorithm, the ADAM, the LBFGS amoung others [4, 11]. In this work we will use the ADAM optimizer since it is frequently used in the literature as a default optimization algorithm for deep learning and it is recommended for most of the applications. In addition, the algorithm is straightforward to implement, has fast computation time, requires few parameters for tuning, has low memory requirements and can handle noisy sparse gradients [16].

The initialization of the weights and biases of the neural networks will be considered in the numerical results.

The procedure to stop the algorithm will be the early stopping [11]. We spilt the data in train and validation set and we run the algorithm until the error on the validation set has not improved for a number of iterations. This specific number of iterations is called the patience in the early stopping.

Algorithm 1 sketches the combination of the backpropagation technique with the ADAM optimizer where the gradient vector in (29) is computed using the information in (27) and (28). As presented, is usually called ADAM with mini-batch training.

3.3 Analysis of the optimization method

Here we establish some bounds for the relative error between the exact solution and the approximation given by the neural network. Let us assume that we are able to minimize the objective function (16) so that the absolute error for each term is bounded by ϵ_i^n, that is,

$$
\left\| A_i^n \left[\boldsymbol{\mu}_{\boldsymbol{\theta}} \; \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal} - T_i^n \right\|_\infty \leq \epsilon_i^n, \quad i \in \{1, 2, \dots, Q\}, n \in \{1, 2, \dots, T\}. \tag{30}
$$

```
Algorithm 1 Backpropagation with ADAM optimizer
initialization Collect the points $\mathbf{x}_p = (x_{1p}, x_{2p}, x_{3p})^\intercal$, $p \in \{1, 2, ..., M\}$ and split the data $A_i^n, T_i^n$, $i \in \{1, ..., Q\}$, $n \in \{1, 2, \dots, T\}$ in the train and in the validation sets. Consider the $L$-layer neural networks $\mu_{\boldsymbol{\theta}}$ and $\lambda_{\boldsymbol{\theta}}$ defined in (18) and compute the initial parameters $\theta_\mu$, $\theta_\lambda$. Initialize the first and second moments to zero: $\mathbf{M}_\mu = 0$, $\mathbf{V}_\mu = 0$, $\mathbf{M}_\lambda = 0$, $\mathbf{V}_\lambda = 0$. Define the constants: learning rate $\alpha_\mu$, $\alpha_\lambda$, exponential decay rates for moment estimation $\beta_1$ , $\beta_2 \in [0, 1[$, $\varepsilon$ used for numerical stabilization (small number), the exponential decay $\gamma$ and the sample size $q$. Note: in the algorithm $\odot$ denotes the element-wise product.
while stopping criterion not met do
    Sample a mini-batch of $q$ examples from the training set $A_i^n, T_i^n$, $i = 1, ..., i^*$, $n = 1, ..., n^* : i^* + n^* = q$.
    Compute the predictions $\mu_{\boldsymbol{\theta}}$ and $\lambda_{\boldsymbol{\theta}}$.
    Compute the gradient $G = \frac{1}{q}\sum_i \sum_n \nabla \hat{L}_{\boldsymbol{\theta}}$ using the gradient vector $\nabla \hat{L}_{\boldsymbol{\theta}}$ in (29).
    Define $k = k + 1$.
    Update biased first moment estimate:
    $\mathbf{M}_\mu = \beta_1 \mathbf{M}_\mu + (1 - \beta_1)\, G$,    $\mathbf{M}_\lambda = \beta_1 \mathbf{M}_\lambda + (1 - \beta_1)\, G$;
    Update biased second moment estimate:
    $\mathbf{V}_\mu = \beta_2 \mathbf{V}_\mu + (1 - \beta_2)\, G \odot G$,   $\mathbf{V}_\lambda = \beta_2 \mathbf{V}_\lambda + (1 - \beta_2)\, G \odot G$ ;
    Correct bias in first moment:
    $\widehat{\mathbf{M}}_\mu = \mathbf{M}_\mu / \left(1 - \beta_1^k\right)$,    $\widehat{\mathbf{M}}_\lambda = \mathbf{M}_\lambda / \left(1 - \beta_1^k\right)$;
    Correct bias in second moment:
    $\widehat{\mathbf{V}}_\mu = \mathbf{V}_\mu / \left(1 - \beta_2^k\right)$,    $\widehat{\mathbf{V}}_\lambda = \mathbf{V}_\lambda / \left(1 - \beta_2^k\right)$;
    Update $\theta_\mu$ and $\theta_\lambda$ with the formulas:
    $\theta_\mu = \theta_\mu - \alpha_\mu \frac{\widehat{\mathbf{M}}_\mu}{\sqrt{\widehat{\mathbf{V}}_\mu + \varepsilon}}$ and $\theta_\lambda = \theta_\lambda - \alpha_\lambda \frac{\widehat{\mathbf{M}}_\lambda}{\sqrt{\widehat{\mathbf{V}}_\lambda + \varepsilon}}$ respectively;
    Apply the exponential decay $\gamma$ to $\alpha_\mu$ and $\alpha_\lambda$:
    $\alpha_\mu = \alpha_\mu \gamma$,   $\alpha_\lambda = \alpha_\lambda \gamma$.
return Parameters $\theta_\mu$ and $\theta_\lambda$.
```

Note that we are always able to calculate the value of $\left\| A_i^n \left[\boldsymbol{\mu_\theta}\ \boldsymbol{\lambda_\theta}\right]^\intercal - T_i^n \right\|_\infty^2$ in the end of the process.

First we are going to find an estimate for the relative error between $\left[\boldsymbol{\mu}\ \boldsymbol{\lambda}\right]^\intercal$ and $\left[\boldsymbol{\mu_\theta}\ \boldsymbol{\lambda_\theta}\right]^\intercal$ assuming that the set of data $\mathbf{u}_{ih}^{n-1}, \mathbf{u}_{ih}^{n}, \mathbf{u}_{ih}^{n+1}, \mathbf{f}_i^n, \mathbf{g}_i^n$ satisfies $A_i^n \left[\boldsymbol{\mu}\ \boldsymbol{\lambda}\right]^\intercal = T_i^n$. Defining $(A_i^n)^{-1} = ((A_i^n)^\intercal A_i^n)^{-1} (A_i^n)^\intercal$ the pseudoinverse matrix of A_i^n and $\mathrm{cond}_\infty(A_i^n) = \left\| (A_i^n)^{-1} \right\|_\infty \| A_i^n \|_\infty$ the conditioning number of the matrix A_i^n, we have the following result [20],

$$\frac{\left\| \left[\boldsymbol{\mu} - \boldsymbol{\mu_\theta}\ \ \boldsymbol{\lambda} - \boldsymbol{\lambda_\theta}\right]^\intercal \right\|_\infty}{\left\| \left[\boldsymbol{\mu}\ \boldsymbol{\lambda}\right]^\intercal \right\|_\infty} \leq \frac{\mathrm{cond}_\infty(A_i^n)}{\|T_i^n\|_\infty} \epsilon_i^n. \tag{31}$$

The estimate (31) is usually not very sharp. We can write

$$\begin{aligned}\left[\boldsymbol{\mu}-\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}-\boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal} &= (A_i^n)^{-1}T_i^n-(A_i^n)^{-1}A_i^n\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\\ &= (A_i^n)^{-1}\left(T_i^n-A_i^n\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right).\end{aligned}$$

Applying norms and using (30) we obtain

$$\frac{\left\|\left[\boldsymbol{\mu}-\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}-\boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}\leq\frac{\|(A_i^n)^{-1}\|_\infty}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}\epsilon_i^n.$$

Involving $|\Lambda|\,Q$ samples of data set $\mathbf{u}_{ih}^{n-1},\mathbf{u}_{ih}^{n},\mathbf{u}_{ih}^{n+1},\mathbf{f}_i^n,\mathbf{g}_i^n,\ i\in\{1,...,Q\},n\in\{1,2,\ldots,T\}$ we get the average

$$\frac{\left\|\left[\boldsymbol{\mu}-\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}-\boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}\leq\frac{1}{|\Lambda|\,Q}\sum_{n\in\Lambda}\sum_{i=1}^{Q}\frac{\|(A_i^n)^{-1}\|_\infty}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}\epsilon$$

where $\epsilon_i^n\leq\epsilon,\forall i\in\{1,2,\ldots,Q\},\forall n\in\{1,2,\ldots,T\}$. In this way we proved the following result.

Theorem 1. *Let us consider QT samples of data sets of the form $\mathbf{u}_{ih}^{n-1},\mathbf{u}_{ih}^{n}$, $\mathbf{u}_{ih}^{n+1},\mathbf{f}_i^n,\mathbf{g}_i^n,\ i\in\{1,...,Q\},\ n\in\{1,2,\ldots,T\}$ which satisfy $A_i^n\left[\boldsymbol{\mu}\ \boldsymbol{\lambda}\right]^{\intercal}=T_i^n$. Let us assume that A_i^n a non singular matrix. Suppose $\epsilon_i^n\leq\epsilon,\forall i\in\{1,...,Q\},\forall n\in\{1,2,\ldots,T\}$ then*

$$\frac{\left\|\left[\boldsymbol{\mu}-\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}-\boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}\leq\frac{1}{|\Lambda|\,Q}\sum_{n\in\Lambda}\sum_{i=1}^{Q}\frac{\|(A_i^n)^{-1}\|_\infty}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_\infty}\epsilon.$$

In real applications, the experimental data contains noise. Therefore our next task will be to derive a similar result of Theorem 1 for noisy data.

If we consider noise in the displacements then, by (13), the data $\mathbf{u}_{ih}^n$, $i\in\{1,...,Q\}$, $n\in\{1,2,\ldots,T\}$ will contain noise. Therefore the matrix A_i^n and the vector T_i^n will contain errors as well by (14) and (15). We denote the matrix and the vector affected by noise as $\bar{A}_i^n$ and $\bar{T}_i^n$ respectively.

The errors associated are given by $\Delta A_i^n=A_i^n-\bar{A}_i^n$ and $\Delta T_i^n=T_i^n-\bar{T}_i^n$ where the errors ΔA_i^n and ΔT_i^n are a matrix and vector whose entries are the absolute errors of the entries in A_i^n and T_i^n respectively. Defining $R_i^n=\bar{T}_i^n-\bar{A}_i^n\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}$ then, as previously, we can obtain the bound $\|R_i^n\|_\infty^2\leq\epsilon_i^n$. First observe that

$$\left(A_i^n-\Delta A_i^n\right)\left[\boldsymbol{\mu}-\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}-\boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}=R_i^n+\Delta T_i^n-\Delta A_i^n\left[\boldsymbol{\mu}\ \boldsymbol{\lambda}\right]^{\intercal}. \tag{32}$$

By (32) we also can get

$$\left[\boldsymbol{\mu}-\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}-\boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}=(A_i^n)^{-1}\left(R_i^n+\Delta T_i^n-\Delta A_i^n\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right),$$

and then

$$\frac{\left\|\left[\boldsymbol{\mu}-\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}-\boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_{\infty}}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_{\infty}} \leq \frac{\|(A_i^n)^{-1}\|_{\infty}}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_{\infty}}\epsilon_i^n + \frac{\|(A_i^n)^{-1}\Delta T_i^n\|_{\infty}}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_{\infty}} + \|(A_i^n)^{-1}\Delta A_i^n\|_{\infty}.$$

Theorem 2. *Let us consider QT samples of data sets of the form* $\mathbf{u}_{ih}^{n-1}, \mathbf{u}_{ih}^{n}, \mathbf{u}_{ih}^{n+1}, \mathbf{f}_i^n, \mathbf{g}_i^n$, $i \in \{1, ..., Q\}$, $n \in \{1, 2, \dots, T\}$. *Let us assume that* A_i^n *is a non singular matrix. Assuming* $\epsilon_i^n \leq \epsilon$, $\forall i \in \{1, 2, \dots, Q\}$, $\forall n \in \{1, 2, \dots, T\}$, *then*

$$\frac{\left\|\left[\boldsymbol{\mu}-\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}-\boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_{\infty}}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_{\infty}} \leq \frac{1}{|\Lambda|\,Q}\sum_{n\in\Lambda}\sum_{i=1}^{Q}\frac{\|(A_i^n)^{-1}\|_{\infty}}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_{\infty}}\epsilon + \frac{1}{|\Lambda|\,Q}\sum_{n\in\Lambda}\sum_{i=1}^{Q}\frac{\|(A_i^n)^{-1}\Delta T_i^n\|_{\infty}}{\left\|\left[\boldsymbol{\mu}_{\boldsymbol{\theta}}\ \boldsymbol{\lambda}_{\boldsymbol{\theta}}\right]^{\intercal}\right\|_{\infty}} + \frac{1}{|\Lambda|\,Q}\sum_{n\in\Lambda}\sum_{i=1}^{Q}\|(A_i^n)^{-1}\Delta A_i^n\|_{\infty}. \tag{33}$$

Observe that although the estimate (33) cannot be calculated in practice, it measures the impact of the noise.

4 Numerical results

In this section we present some computational results to evaluate the applicability of the proposed method.

In the tests performed we used fabricated data obtained by simulating the direct problem and for observed displacements U^n, we consider the solution of (10). We will consider two scenarios in simulations, namely the cases of homogeneous and heterogeneous media: with λ and μ constants to compare our approach with the methodology proposed in [2] and with λ space dependent to mimic possible heterogeneity in different layers of the medium [19]. For both scenarios we consider the following setting: $\Omega = [-2, 2]^3$ with $\partial\Omega = \Gamma_1 \cup \Gamma_2$ where Γ_1 is the face of the cube contained in the plane $x_3 = -2$; the mesh is a partition of Ω into 162 tetrahedrons; $\rho = 1\text{g}/cm^3$; the functions $\mathbf{g}_1^n$ and $\mathbf{f}_1^n$ are defined respectively by $\mathbf{g}_1^n = (0,\ 0,\ \cos(t^n) \times 10^5)$ and $f_1^n = 0$. In the homogeneous medium, $(\mu(x), \lambda(x)) = (1.6069 \times 10^6, 1.4462 \times 10^7)$ and, in the heterogeneous medium, $\mu(x) = 1.6069 \times 10^6$ and

$$\lambda(x) = \begin{cases} 1.5 \times 10^7 & , \text{ if } x_3 \geq 0 \\ 1.4 \times 10^7 & , \text{ if } x_3 < 0 \end{cases}.$$

We performed experiments with noise free data as well as noisy data in order to assess the sensitivity of our method to noise. To check the robustness of the proposed method when considering noisy data we consider Gaussian noise $R \sim N(0, \sigma)$ where R is a random vector of dimension $3N \times 1$ and σ is the standard deviation. So instead U^n, we consider as data $\bar{U}^n = (R + 1_{3N\times 1})U^n$, where $1_{3N\times 1}$ is a $3N \times 1$ vector with all components equal to one and the i-th component of the vector $\bar{U}^n$ is given by $(R(i) + 1)U^n(i)$, $i \in \{1, ..., 3N\}$.

In simulations of noise free data we will use a single data set in space, that is $Q = 1$, and all information in time, $T = 1000$. Therefore, in Algorithm 1, the mini-batch of q examples will have $i^* = 0$ so $n^* = q$ which means all the data of the mini-batch correspond to q different time steps. In noisy data, fixing the level of noise, we generate randomly 100 data sets so $Q = 100$ and $T = 1000$.

In both scenarios the time step is given by $t^n = n\Delta t$, $n = 0, 1, \cdots, 1001$, with $\Delta t = 1.2 \times 10^{-5}$. In Algorithm 1 we use $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-4}$ (the parameters for which we achieved better results in our tests) and the division of data A_i^n, T_i^n, $i \in \{1, ..., Q\}$, $n \in \{1, 2, \ldots, T\}$ is 82% for training and 18% for the validation set.

In this work we consider variations of σ in the set $\{0, 10^{-7}, 10^{-5}, 10^{-3}, 10^{-1}\}$.

4.1 Homogeneous medium

We start to analyze the method when the optimal solution is given by

$$(\mu(x), \lambda(x)) = (1.6069 \times 10^6, 1.4462 \times 10^7). \tag{34}$$

Since the optimal solution is constant, the neural networks will be given by:

$$\mu_{\boldsymbol{\theta}}(\mathbf{x}_p) = \mathbf{b}^2 \text{and } \lambda_{\boldsymbol{\theta}}(\mathbf{x}_p) = \mathbf{d}^2.$$

Figure 2 present the scheme of feed-forward neural networks to obtain the outputs $\mu_{\boldsymbol{\theta}}(\mathbf{x}_p)$ and $\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p)$ respectively.

Considering this setting, we performed five simulations using different starting points in the set $I = [0.9\mu, 1.1\mu] \times [0.9\lambda, 1.1\lambda]$ for $\mu = 1.6069 \times 10^6$ and $\lambda = 1.4462 \times 10^7$. The biases $\mathbf{b}^2$ and $\mathbf{d}^2$ follow an uniform distribution, that is, $\mathbf{b}^2 \sim \mathcal{U}([0.9\mu, 1.1\mu])$ and $\mathbf{d}^2 \sim \mathcal{U}([0.9\lambda, 1.1\lambda])$. We also use the following set of hyperparameters: learning rate $\alpha_\mu = \alpha_\lambda = 10^5$ with exponential decay $\gamma = 0.936$ and mini-batch size $q = 32$. For the early stopping we define a patience of 7.

Figure 3 presents the relative error average in parameters and the corresponding relative error average in displacements $\|U - U_{obs}\|^2_{L^2_h(\Omega)} / \|U_{obs}\|^2_{L^2_h(\Omega)}$, obtained from five simulations for each value of σ. As we can see the optimal solution is globally well recovered where the relative errors in parameters seem to grow along the increase of the noise level in a linear form. Moreover, analysing the relative error in terms of the displacements, we obtained on average relative errors in displacements approximately of the same order comparing to those error in parameters.

In Table 1 we compare the values of the relative error in parameters corresponding to the results in Figure 3 with the results obtained with the methodology described in

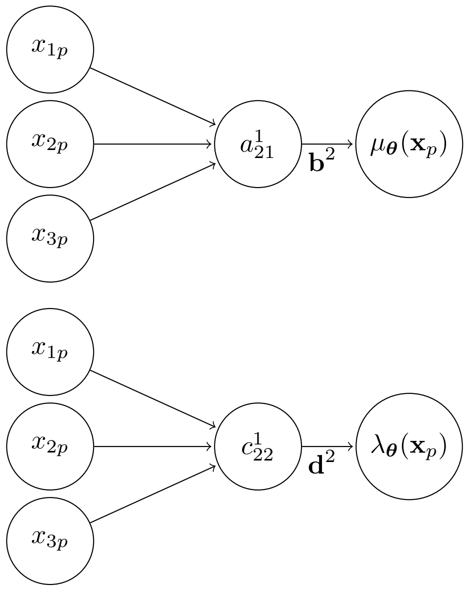


Fig. 2: Scheme of the 1-layer networks to approximate the optimal solution (34).

[2]. For the method in [2], we used the same setting described previously adding the angular frequency $w = 1$ and the functions $\mathbf{g} = (0,\ 0,\ 10^5)$ and $\mathbf{f} = 0$ and the relative error average in parameters was obtained from thirty simulations for each value of σ.

Seeing in detail the values in Table 1, we observe, in general, better results when Algorithm 1 is used, with few exceptions (*e.g.* the case $\sigma = 10^{-1}$). Therefore our method presented based on neural networks seems to be a better option when comparing to the approach studied in [2].

	Parameter	μ		λ	
	Formulation	[2]	Algorithm 1	[2]	Algorithm 1
	0	9.5×10^{-16}	1.3×10^{-15}	5.9×10^{-16}	4.1×10^{-16}
	10^{-7}	7.7×10^{-7}	2.9×10^{-7}	9.8×10^{-7}	1.3×10^{-7}
σ	10^{-5}	6.7×10^{-5}	4.1×10^{-5}	8.0×10^{-5}	9.3×10^{-6}
	10^{-3}	5.7×10^{-3}	1.1×10^{-3}	7.1×10^{-3}	5.1×10^{-4}
	10^{-1}	1.3×10^{-1}	6.5×10^{-1}	1.9×10^{-1}	1.1×10^{-1}

Table 1: Relative errors in parameters corresponding to the method proposed in [2] and to the Algorithm 1.

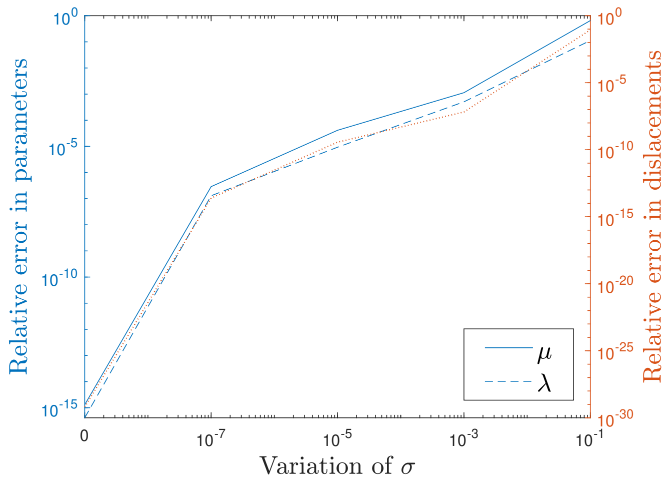


Fig. 3: Relative error average in parameters μ (continuous line in blue) and λ (dashed line in blue) obtained from five simulations for each value of σ. The orange dotted line corresponds to the relative error average in displacements for the same simulations. The results consider the optimal solution in (34) and the Algorithm 1.

4.2 Heterogeneous medium

In the second example, the optimal solution is given by

$$\mu(x) = 1.6069 \times 10^6 \tag{35}$$

and

$$\lambda(x) = \begin{cases} 1.5 \times 10^7 & , \text{ if } x_3 \geq 0 \\ 1.4 \times 10^7 & , \text{ if } x_3 < 0 \end{cases} \tag{36}$$

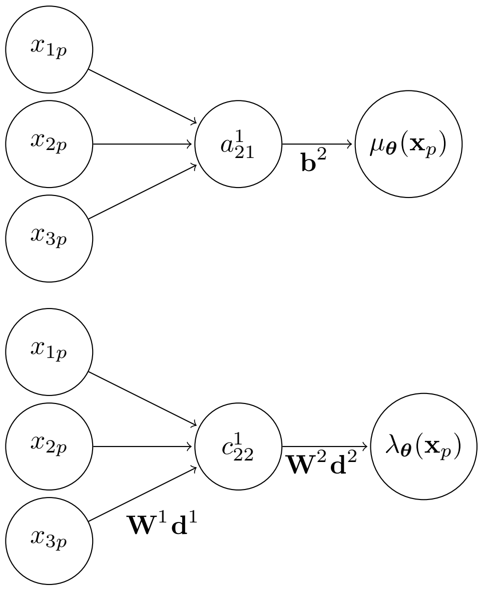
The neural networks take the following form:

$$\mu_{\boldsymbol{\theta}}(\mathbf{x}_p) = \mathbf{b}^2 \text{and } \lambda_{\boldsymbol{\theta}}(\mathbf{x}_p) = \mathbf{W}^2 \sigma^1 \left(\mathbf{W}^1 \mathbf{x}_p + \mathbf{d}^1 \right) + \mathbf{d}^2,$$

with $\sigma^1(x) = \dfrac{1}{1+e^{-x}}$ being the sigmoid activation function. In Figure 4 we present the schemes of feed-forward neural networks.

In terms of initialization of the parameters we consider $\mathbf{b}^2 \sim \mathcal{U}([0.9\mu, 1.1\mu])$, $\mathbf{d}^1 \sim \mathcal{U}([-1, 1])$, $\mathbf{d}^2 \sim \mathcal{U}([0.9\lambda, 1.1\lambda])$, $\mathbf{W}^1 \sim \mathcal{U}\left(\left[-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right]\right)$ and $\mathbf{W}^2 \sim \mathcal{U}([0, 0.2\lambda])$, with $\mu = 1.6069 \times 10^6$ and $\lambda = 1.4462 \times 10^7$, since they are suitable with the function we want to approximate.

In the simulations we noticed that the learning rate α_λ must change depending on the parameter. For that reason we will call $\alpha_{\mathbf{d}^1}$, $\alpha_{\mathbf{d}^2}$, $\alpha_{\mathbf{W}^1}$ and $\alpha_{\mathbf{W}^2}$ the learning rate for the parameters $\mathbf{d}^1$, $\mathbf{d}^2$, $\mathbf{W}^1$ and $\mathbf{W}^2$, respectively. We defined $\alpha_\mu = 10^5$, $\alpha_{\mathbf{d}^1} = 0.25$,



x_{1p}
x_{2p}
x_{3p}
a^1_{21}
$\mathbf{b}^2$
$\mu_{\boldsymbol{\theta}}(\mathbf{x}_p)$
x_{1p}
x_{2p}
x_{3p}
c^1_{22}
$\mathbf{W}^2\mathbf{d}^2$
$\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p)$
$\mathbf{W}^1\mathbf{d}^1$

Fig. 4: Scheme of the 1-layer networks to approximate the optimal solution (35)-(36).

$\alpha_{\mathbf{d}^2} = 8 \times 10^4$, $\alpha_{\mathbf{W}^1} = 1.5$ and $\alpha_{\mathbf{W}^2} = 5 \times 10^4$. In Algorithm 1 the exponential decay is $\gamma = 0.99975$, the size of mini-batch is $q = 320$ and the early stopping is defined with a patience of 700.

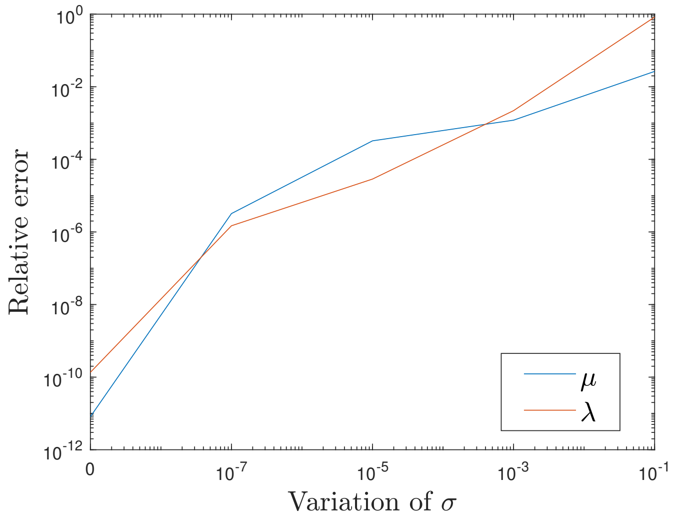


Fig. 5: Relative error average obtained from five simulations for each value of σ. The results consider the optimal solution in (35)-(36) and the Algorithm 1.

Figure 5 shows the evolution of the relative error average obtained from five simulations for each value of σ. The relative error is measured in the infinite norm,

$$\frac{\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_{\boldsymbol{\theta}}\|_\infty}{\|\boldsymbol{\lambda}\|_\infty} = \frac{\max\limits_{1\leq p\leq M} |\lambda_{\boldsymbol{\theta}}(\mathbf{x}_p) - \lambda(\mathbf{x}_p)|}{\max\limits_{1\leq p\leq M} |\lambda(\mathbf{x}_p)|}.$$

As expected, increasing the complexity of the neural network rises the challenge to obtain the optimal solution but the results are promising.

Funding

This work was partially supported by the Centre for Mathematics of the University of Coimbra (funded by the Portuguese Government through FCT/MCTES, DOI 10.54499/UIDB/00324/2020); the FEDER Funds through the Operational Program for Competitiveness Factors - COMPETE and by Portuguese National Funds through FCT - Foundation for Science and Technology, DOI 10.54499/2021.06672.BD; the FCT (Portugal) research project PTDC/EMD-EMD/32162/2017, COMPETE and Portugal 2020.

Declarations

Competing interests: The authors declare no conflict of interest.

Author's contributions: Conceptualization, R.H. and S.B. ; methodology, R.H. and S.B.; formal analysis, R.H. and S.B.; software, R.H.; validation, R.H. and S.B.; supervision, S.B.; writing original draft preparation, R.H.; writing-review and editing, R.H. and S.B.. All authors have read and agreed to the published version of the manuscript.

Availability of data: The code supporting the conclusions of this manuscript will be made available by the authors upon formal and reasonable request.

Ethics Approval: The authors declare no approval committee.

References

[1] M. Ainsworth and C. Parker. Unlocking the secrets of locking: Finite element analysis in planar linear elasticity. Computer Methods in Applied Mechanics and Engineering, 395, 2022.

[2] S. Barbeiro, R. Henriques and J. Santos. A quadratic optimization program for the inverse elastography problem. Journal of Mathematics in Industry 14, article number:18, https://doi.org/10.1186/s13362-024-00156-7, 2024.

[3] S. Barbeiro and R. Henriques, J. Santos. The derivative free trust-region method for the inverse elastography problem. Accepted for publication in Proceedings of the 22nd ECMI Conference on Industrial and Applied Mathematics, Springer, 2023.

[4] C. M. Bishop. Neural Networks for Pattern Recognition, Department of Computer Science and Applied Mathematics Aston University Birmingham, UK. Clarendon press, Oxford, 1995.

[5] C. Carstensen, G. Dolzmann, S. A. Funken and D. S. Helm, Locking-free adaptive mixed finite element methods in linear elasticity. Computer Methods in Applied Mechanics and Engineering, 190 (13–14), 1701-1718, 2000.

[6] D. Claus, M. Mlikota, J. Geibel, T. Reichenbach, G. Pedrini, J. Mischinger, S. Schmauder and W. Osten. Large-field-of-view optical elastography using digital image correlation for biological soft tissue investigation. Journal of Medical Imaging, 4 (1): 1–14, 2017.

[7] M. M. Doyley. Model-based elastography: a survey of approaches to the inverse elasticity problem. Physics in Medicine and Biology, 57 (3):R35-R73, 2012.

[8] R. Fitzpatrick. Theoretical Fluid Mechanics, IOP Publishing, 978-0-7503-1552-4, 2018.

[9] J. Ghaboussi and D. Sidarta. New nested adaptive neural networks (NANN) for constitutive modeling. Computers and Geotechnics, vol. 22, 29–52, 1998.

[10] G. Giannakoulas , G. Giannoglou, J. Soulis, T. Farmakis, S. Papadopoulou, G. Parcharidis and G. Louridas. A computational model to predict aortic wall stresses in patients with systolic arterial hypertension. Elsevier, Medical Hypotheses, 2005.

[11] I. Goodfellow, Y. Bengio and A. Courville. Deep Learning, pre-pub version, MIT Press, 2016.

[12] M. T. Hagan, H. B. Demuth, M. H. Beale and O. D. Jesús. Neural Network Design, 2nd edition, 9780971732117, 2014.

[13] E. Haghighat, M. Raissi, A. Moure, H. Gomez and R. Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Journal of Computer methods in applied mechanics and enginnering, 379, 113741, 2021. URL: https://doi.org/10.1016/j.cma.2021.113741.

[14] K. Hornik. Approximation capabilities of multilayer feedforward networks, Neural networks, vol. 4, no. 2, pp. 251–257, 1991.

[15] B. F. Kennedy, X. Liang, S. G. Adie, D. K. Gerstmann, B. C. Quirk, S. A. Boppart and D. D. Sampson. In vivo three-dimensional optical coherence elastography. Opt. Express, 19 (7):6623–6634, 2011.

[16] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization, arXiv, 2017.

[17] J. C. Mason and David C. Handscomb. Chebyshev Polynomials. Computer Science, Mathematics and Statistics, 2002.

[18] E. Park and A. M. Maniatty. Shear modulus reconstruction in dynamic elastography: time Harmonic case. Phys. Med. Biol. 51(15), 3697-3721, 2006.

[19] Y. Qu, Y. He, Y. Zhang, T. Ma, J. Zhu, Y. Miao, C. Dai, M. Humayun, Q. Zhou and Z. Chen. Quantified elasticity mapping of retinal layers using synchronized acoustic radiation force optical coherence elastography. Biomed.Opt.Express, 9 (9):4054–4063, 2018.

[20] A. Quarteroni, R. Sacco and F. Saleri, Numerical Mathematics, Springer, 2000.

[21] M. Raissi, P. Perdikaris and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378, 686–707, 2019. URL: https://doi.org/10.1016/j.jcp.2018.10.045. doi:10.1016/j.jcp.2018.10.045.

[22] P. A. Raviart and J.M. Thomas. Introduction à l'analyse numerique des équations aux dérivées partielles, Masson, 1983.

[23] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. Advances in Computational Mathematics, 11(2-3):193–210, 1999.

[24] S. Rudy, A. Alla, S. L. Brunton and J. N. Kutz. Data-driven identification of parametric partial differential equations. SIAM Journal on Applied Dynamical Systems 18, 643–660, 2019. URL: https://epubs.siam.org/doi/abs/10.1137/18M1191944. doi:10.1137/18M1191944.

[25] U. Saravanan. Advanced Solid Mechanics. Traction and Stress, 2013.

[26] P. Serranho, S. Barbeiro, R. Henriques, A. Batista, M. Santos, C. Correia, J. Domingues, C. Loureiro, J. Cardoso, R. Bernardes and M. Morgado. On the Numerical Solution of the Inverse Elastography Problem for Time-harmonic Excitation. In Proceedings of the 2nd International Conference on Image Processing and Vision Engineering (Improve 2022), pages 259-264, 2022.

[27] K. S. Sheinerman and S. R. Umansky, Circulating cell-free microRNA as biomarkers for screening, diagnosis and monitoring of neurodegenerative diseases and other neurologic pathologies. Front. Cell. Neurosci, vol. 7, 2013.

[28] J. Wang, Y. Xu and S. A. Boppart. Review of optical coherence tomography in oncology. J Biomed Opt, 22(12): 121711, 2017.

[29] J. Zhu, Y. Miao, L. Qi, Y. Qu, Y. He, Q. Yang and Z. Chen. Longitudinal shear wave imaging for elasticity mapping using optical coherence elastography. Applied Physics Letters, 110 (20):201101, 2017.

[30] Y. Zhu, N. Zabaras and P. S. Koutsourelakis, P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Journal of Computational Physics 394, 56–81, 2019

. URL: https://www.sciencedirect.com/science/article/pii/S0021999119303559. doi:10.1016/j.jcp.2019.05.024.