INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2008; **75**:1341–1360
Published online 19 February 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.2304

Neural networks for variational problems in engineering

R. Lopez[1, *, †], E. Balsa-Canto[2] and E. Oñate[1]

[1]*International Center for Numerical Methods in Engineering (CIMNE), Edificio C1, Gran Capitán s/n, 08034 Barcelona, Spain*
[2]*Process Engineering Group, IIM-CSIC, Spanish Council for Scientific Research, Eduardo Cabello 6, 36208 Vigo, Spain*

SUMMARY

In this work a conceptual theory of neural networks (NNs) from the perspective of functional analysis and variational calculus is presented. Within this formulation, the learning problem for the multilayer perceptron lies in terms of finding a function, which is an extremal for some functional. Therefore, a variational formulation for NNs provides a direct method for the solution of variational problems.

This proposed method is then applied to distinct types of engineering problems. In particular a shape design, an optimal control and an inverse problem are considered. The selected examples can be solved analytically, which enables a fair comparison with the NN results. Copyright © 2008 John Wiley & Sons, Ltd.

Received 13 December 2006; Revised 29 October 2007; Accepted 13 December 2007

KEY WORDS: neural networks; multilayer perceptron; functional analysis; variational calculus; shape design; optimal control; inverse problems

1. INTRODUCTION

Many problems arising in science and engineering aim to find a function that is the minimal or the maximal value of a specified functional. These type of problems are regarded as variational problems [1].

Shape design [2], optimal control [3] or inverse problems [4] fall into the general class of variational problems, and they are considered of great engineering interest. A typical shape design problem in the aeronautical industry is that of finding airfoil sections with reduced drag [5]. One example of optimal control in the chemical sector is to determine the feed rate that maximizes the yield in a chemical fermenter [6]. A significant inverse problem in the metallurgy industry is estimating the thermal properties of a melting metal [7].

*Correspondence to: R. Lopez, International Center for Numerical Methods in Engineering (CIMNE), Edificio C1, Gran Capitán s/n, 08034 Barcelona, Spain.
†E-mail: rlopez@cimne.upc.edu

However, while some variational applications can be solved analytically, in most practical cases the only general technique is to approximate the solution using direct methods. The fundamental idea is to consider the variational problem as a limit problem for some function optimization problem in many dimensions [8].

The direct methods of Euler and Ritz have been widely applied and are well covered in many textbooks. In the Euler method, the values of the functional are considered along piecewise linear functions, whereas in the Ritz method they are considered along some linear combination of functions. Along such curves the functional turns into a function of some coefficients. The issue is then to find the coefficients that minimize or maximize that function by solving a function optimization problem. Alternative techniques based on Laguerre polynomials [9], Legendre polynomials [10], Chebyshev polynomials [11] or more recent wavelets [12], for instance, have also been proposed.

Unfortunately, variational problems might be extremely difficult to solve in practice. The presence of non-linear and dynamic constraints, the deficient approximation or convergence properties of the selected basis functions, the usual large dimension in the resulting function optimization problem, the appearance of local optima or solutions presenting oscillatory behaviors are some of the most typical complications.

In that concern, here we present a conceptual theory of neural networks (NNs) from the perspective of functional analysis and variational calculus, which aims to overcome such difficulties.

An NN is a biologically inspired computational model, which consists of a network architecture composed of artificial neurons [13]. This structure contains a set of free parameters, which can be adjusted to perform certain tasks.

The multilayer perceptron (MLP) is an important NN model, and much of the literature in the field refer to it. Traditionally, the learning problem for the MLP has been stated in terms of the minimization of an error function of the free parameters, in order to fit the NN outputs to an input-target data set [14]. In this way, the learning tasks allowed belong to the category of data modeling, such as function regression, pattern recognition or time series prediction.

Regarding engineering applications, NNs are nowadays being used to solve a whole range of problems [15]. Most common case studies include modeling hitherto intractable processes, designing complex feed-back control signals, building meta-models for posterior optimization or detecting device faults. All that applications need from an input-target data set in order to evaluate the error function and therefore fall into one of the three categories of data modeling enumerated above.

In this work a variational formulation for the MLP is presented. Within this formulation, learning means to solve a variational problem by minimizing or maximizing an objective functional associated with the NN. The choice of a suitable objective functional depends on the particular application, and its evaluation may need the integration of functions, ordinary differential equations (ODEs) or partial differential equations (PDEs).

As it will be shown, this formulation provides a direct method for solving variational problems including function regression, pattern recognition, time series prediction, shape design, optimal control or inverse problems, among many others. Remark that the universal approximation properties [16] cause neural computation to be a very appropriate paradigm for that purpose.

Previous research made already some efforts in this direction. Zoppoli *et al.* [17] discussed the approximation properties of different classes of NNs as linear combinations of non-fixed basis functions, and applied this theory to some stochastic functional optimization problems. In [18], Sarkar and Modak make use of neurocomputing to determine several optimal control profiles for different chemical reactors. Also, Franco-Lara and Weuster-Botz [19] estimated optimal feeding strategies for bed-batch bioprocesses using NNs.

This study complements and generalizes that research, and introduces a number of novel issues:

(i) In Section 2, a conceptual theory of NNs from a variational point of view is written. In particular, we introduce the idea of the function space spanned by an MLP. This computational tool can also be extended so as to include independent parameters, boundary conditions and bounds. On the other hand, learning tasks are here stated as variational problems, defined by an objective functional that has an associated parameterized function. In addition, we explain the solution approach of the reduced function optimization problem by means of the training algorithm.

(ii) The proposed formulation is general enough to incorporate a large number of variational applications such as shape design, optimal control and inverse problems, which are explained in detail in Sections 3, 4 and 5, respectively. Moreover, the embed of different numerical methods, such as the Simpson method, the Runge–Kutta–Fehlberg method and the finite element method into the objective functional is considered. Every problem class is accompanied by an illustrative example. The selected applications are easy to understand and may be solved analytically, thus allowing a fair analysis of the results provided by the variational MLP.

2. A VARIATIONAL FORMULATION FOR THE MLP

In this section a conceptual theory of the MLP from a variational point of view is presented. Figure 1 depicts a state diagram for the learning problem in that NN. The solving approach here consists of three steps. The first step is to choose a suitable parameterized function space in which the solution to the problem is to be approximated. The elements of this family of functions are those spanned by an MLP. In the second step the variational problem is formulated by selecting an appropriate objective functional, defined on the function space chosen before. The third step is

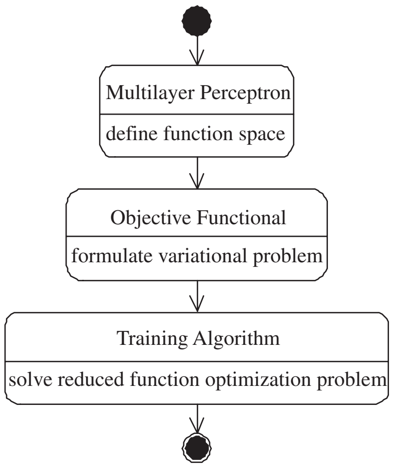

Figure 1. State diagram for the learning problem in the multilayer perceptron.

to solve the reduced function optimization problem. This is performed with a training algorithm capable of finding an optimal set of free parameters.

Within a variational formulation, this NN is described by four elements: a neuron model, the perceptron, a network architecture, the feed-forward, and associated objective functionals and training algorithms.

2.1. *The perceptron neuron model*

The perceptron is the characteristic neuron model in the MLP [20]. It computes a net input signal u as a function h of the input signals $\mathbf{x}$ and the bias and synaptic weights $(b, \mathbf{w})$. The net input signal is then subject to an activation function g to produce an output signal y [13]. Two of the most popular activation functions are the sigmoid function, $g(u) = \tanh(u)$, and the linear function, $g(u) = u$ [21]. Figure 2 shows a perceptron with n inputs.

Mathematically, a perceptron neuron model spans a parameterized function space V from an input $X \subset \mathbb{R}^n$ to an output $Y \subset \mathbb{R}$ [22]. Elements of V are parameterized by the bias and the synaptic weights of the neuron $(b, \mathbf{w})$. Thus, the dimension of V is $n+1$. The elements of the function space, which a perceptron can define, are of the form

$$\begin{aligned} y : \mathbb{R}^n &\to \mathbb{R} \\ \mathbf{x} &\mapsto y(\mathbf{x}; b, \mathbf{w}) \end{aligned}$$

where

$$y(\mathbf{x}; b, \mathbf{w}) = g\left(b + \sum_{i=1}^{n} w_i x_i\right) \tag{1}$$

Although a single neuron can perform certain simple tasks, the power of neural computing comes from connecting many artificial neurons in a network architecture.

2.2. *The feed-forward network architecture*

The architecture of an NN refers to the number of neurons, their arrangement and connectivity [20]. The characteristic network architecture in the MLP is the so-called feed-forward, which typically

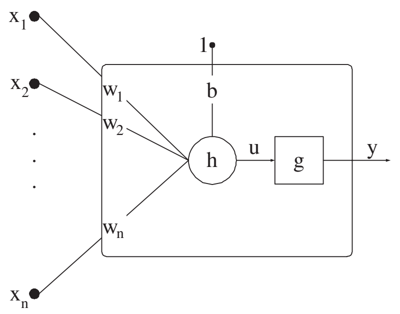

Figure 2. The perceptron neuron model.

consists of a set of sensorial nodes, constituting the input layer, one or more hidden layers of neurons, and another set of neurons, which composes the output layer. Communication proceeds layer by layer from the input layer via the hidden layers up to the output layer [20]. Figure 3 shows an MLP with n inputs, one hidden layer with h_1 neurons and m neurons in the output layer.

As in the case of a single perceptron, an MLP spans a parameterized function space V from an input $X \subset \mathbb{R}^n$ to an output $Y \subset \mathbb{R}^m$ [22]. Elements of V are parameterized by all the biases and synaptic weights in the NN, which can be grouped together in an s-dimensional vector $\underline{\alpha} = (\alpha_1, \ldots, \alpha_s)$. The dimension of V is therefore s. The elements of the function space, which the MLP in Figure 3 can define, are of the form

$$\mathbf{y} : \mathbb{R}^n \to \mathbb{R}^m$$
$$\mathbf{x} \mapsto \mathbf{y}(\mathbf{x}; \underline{\alpha})$$

where

$$y_k(\mathbf{x}; \underline{\alpha}) = g^{(2)} \left(b_k^{(2)} + \sum_{j=1}^{h_1} w_{kj}^{(2)} \cdot g^{(1)} \left(b_j^{(1)} + \sum_{i=1}^{n} w_{ji}^{(1)} x_i \right) \right) \tag{2}$$

for $k = 1, \ldots, m$. This function space can be modified in order to include independent parameters, boundary conditions, lower and upper bounds, etc., as it might be required by the problem.

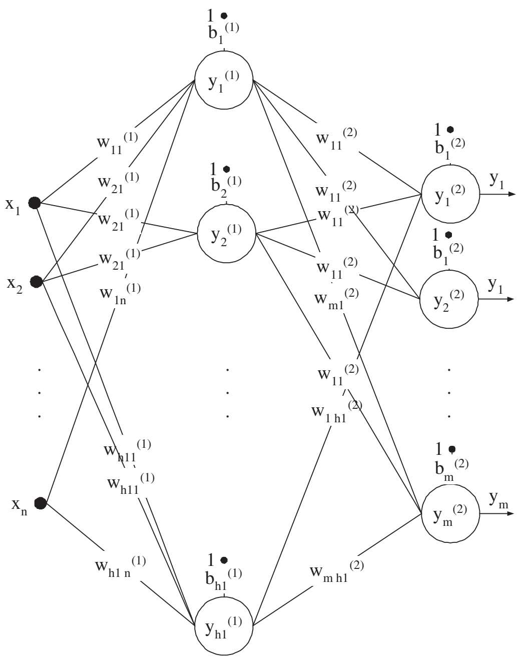

Figure 3. The multilayer perceptron network architecture.

A feed-forward NN with as few as one hidden layer of sigmoid perceptrons and an output layer of linear perceptrons provides a general framework for approximating any function from one finite dimensional space to another up to any desired degree of accuracy, provided sufficiently many hidden neurons are available. In this sense, the MLP is a class of universal approximator [16].

2.3. The objective functional

Traditionally, the learning problem for the MLP has been formulated in terms of the minimization of an error function of the free parameters, in order to fit the NN outputs to an input-target data set [14]. In that way, the only learning tasks allowed are related to modeling of data, such as function regression, pattern recognition or time series prediction.

In a variational formulation, the concept of error function, $e(\underline{\alpha})$, is changed by the concept of objective functional, $F[\mathbf{y}(\mathbf{x};\underline{\alpha})]$ [22]:

$$F : V \to \mathbb{R}$$

$$\mathbf{y}(\mathbf{x};\underline{\alpha}) \mapsto F[\mathbf{y}(\mathbf{x};\underline{\alpha})]$$

The objective functional defines the task that the MLP is required to accomplish and provides a measure of the quality of the representation that it is required to learn. In this way, the choice of a suitable objective functional depends on the particular application. Changing the concept of error function by the concept of objective functional allows us to extend the number of learning tasks for that NN to any variational problem.

The learning problem for the MLP can then be formulated in terms of the optimization of an objective functional of the function space spanned by the NN [22]:

Problem 1 (*Variational problem*)
Let V be the space of all functions $\mathbf{y}(\mathbf{x};\underline{\alpha})$ spanned by an MLP, and let s be the dimension of V. Find an element $\mathbf{y}^*(\mathbf{x};\underline{\alpha}^*) \in V$ for which the functional $F[\mathbf{y}(\mathbf{x};\underline{\alpha})]$, defined on V, takes a minimum or a maximum value.

A variational problem for the MLP can be specified by a set of constraints, which are equalities or inequalities that the solution must satisfy. An easy approach here is to reduce the constrained problem into an unconstrained one by adding a penalty term to the original objective functional for each constraint in the problem. Adding a penalty term gives a large positive or negative value to the objective functional when infeasibility due to a constrain is encountered.

On the other side, the objective functional, $F[\mathbf{y}(\mathbf{x};\underline{\alpha})]$, has an objective function associated with it, $f(\underline{\alpha})$, which is defined as a function of the free parameters in the NN [22],

$$f : \mathbb{R}^s \to \mathbb{R}$$

$$\underline{\alpha} \mapsto f(\underline{\alpha})$$

The minimum or maximum value of the objective functional $F[\mathbf{y}(\mathbf{x};\underline{\alpha})]$ is achieved for a vector of free parameters α^* for which the objective function $f(\underline{\alpha})$ takes a minimum or maximum value, respectively. Therefore, the learning problem for the variational MLP can be reduced to a function optimization problem [22]:

Problem 2 (*Reduced function optimization problem*)
Let $\mathbb{R}^s$ be the space of all vectors $\underline{\alpha}$ spanned by the free parameters of an MLP. Find an element $\underline{\alpha}^* \in \mathbb{R}^s$ for which the function $f(\underline{\alpha})$, defined on $\mathbb{R}^s$, takes a minimum or a maximum value.

Owing to the universal approximation properties, this numerical method would allow to solve applications in any dimension and up to any desired degree of accuracy [22].

2.4. *The training algorithm*

The training algorithm is entrusted to solve the reduced function optimization problem by adjusting the free parameters in the NN so as to optimize the objective function. More specifically, the training algorithm searches in an s-dimensional space for a free parameter vector $\underline{\alpha}^*$ at which the objective function f reaches a minimum or a maximum value.

Training algorithms might require information from the objective function, from the gradient vector and/or from the Hessian matrix of the objective function [23]. These methods, in turn, can perform either global or local optimization tasks.

In that way, zero-order training algorithms make use of the objective function only. The most significant ones are stochastic, which involve randomness in the optimization process. Typical examples are evolutionary algorithms [24] and particle swarm optimization [25], which are able to find global optima. First-order training algorithms, such as conjugate gradient or quasi-Newton methods, use the objective function and its gradient vector. Second-order methods, such as Newton type methods, require Hessian information. In most of the cases, first- and second-order methods have local convergence properties, although some global methods have also been proposed [26].

In this work a quasi-Newton method with BFGS train direction and Brent optimal train rate methods is always selected for training. The tolerance in Brent's method is set to 10^{-6}. On the other hand, the objective function gradient $\nabla f(\underline{\alpha})$ is in all applications computed by means of numerical differentiation [14]. In particular, the central differences method is used, with $\varepsilon = 10^{-6}$.

3. SHAPE DESIGN

The goal in shape design problems is to optimize some performance criterion dependent on a given spatial domain, while satisfying the constraints of the problem [2]. In order to properly define an optimal shape design problem, the following concepts are needed: (i) a mathematical model of the physical system; (ii) a statement of the physical constraints in the final design and (iii) a specification of the performance criterion which measures optimality.

Mathematically, shape design problems are defined as variational problems [2] and their solution, in most of the situations, relies on the use of direct methods [27].

In this work the design of an axisymmetric body of minimum drag [28] is approached by the variational MLP introduced in the previous section.

3.1. *Example: the minimum drag problem*

This is an unconstrained boundary value problem with one input and one output variables, and where the objective functional needs the integration of a function in order to be evaluated. The problem is solved with the recently developed Flood library [29].

3.1.1. Problem statement. Consider the design of a body of revolution with given length l and diameter d providing minimum drag at zero angle of attack. If friction effects are neglected, the drag of such a body with shape $y(x)$ can be expressed as

$$D[y(x)] = 2\pi q \int_0^l y(x)[C_p y'(x)]\,\mathrm{d}x \tag{3}$$

where q is the free-stream dynamic pressure and C_p the pressure coefficient [28]. For a slender body $y'(x) \ll 1$, the pressure coefficient can be approximated by the Newtonian flow relation:

$$C_p = 2[y'(x)]^2 \tag{4}$$

which is valid provided that the inequality $y'(x) \geqslant 0$ is satisfied.

From Equations (3) and (4), the following approximation for the drag is obtained:

$$D[y(x)] = 4\pi q \int_0^l y(x)[y'(x)]^3\,\mathrm{d}x \tag{5}$$

It is convenient to introduce the following dimensionless variables associated with the axial coordinate and the radial coordinate

$$\xi = \frac{x}{l} \tag{6}$$

$$\eta = \frac{2y}{d} \tag{7}$$

Also, a dimensionless coefficient associated with the drag can be defined as

$$C_D[\eta(\xi)] = \tau^2 \int_0^1 \eta(\xi)[\eta'(\xi)]^3\,\mathrm{d}\xi \tag{8}$$

where $\tau = d/l$ is the slenderness of the body.

The analytical solution to the minimum drag problem formulated in this section is given by

$$\eta^*(\xi) = \xi^{3/4} \tag{9}$$

which provides a minimum value for the drag coefficient $C_D[\eta^*(\xi)]/\tau^2 = 0.4220$.

3.1.2. Selection of function space. The body of revolution $\eta(\xi)$, for $\xi \in [0, 1]$, will be represented by an MLP with a sigmoid hidden layer and a linear output layer. This axisymmetric structure is to be written in cartesian coordinates; hence, the NN must have one input and one output neurons. On the other side, an appropriate number of hidden neurons is believed to be 3 for this particular application. This network architecture is depicted in Figure 4.

The output signals from the NN in Figure 4 must be post-processed so as to satisfy the boundary conditions $\eta(0) = 0$ and $\eta(1) = 1$. Such an MLP spans a family V of parameterized functions $\eta(\xi; \underline{\alpha})$ of dimension $s = 10$. Elements of V are of the form

$$\eta : \mathbb{R} \to \mathbb{R}$$

$$\xi \mapsto \eta(\xi; \underline{\alpha})$$

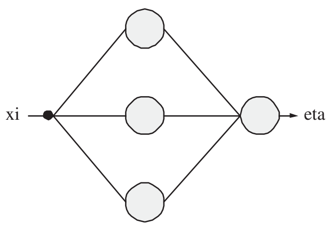

Figure 4. Network architecture for the minimum drag problem.

where

$$\eta(\xi;\underline{\alpha})=\xi+\xi(\xi-1)\left[b_1^{(2)}+\sum_{j=1}^{3}w_{1j}^{(2)}\cdot\tanh(b_j^{(1)}+w_{j1}^{(1)}\xi)\right] \tag{10}$$

The shape function $\eta(\xi)$ is constrained to lie in the interval [0, 1]. To deal with such constraints, the outputs from the NN are bounded in the form

$$\eta(\xi;\underline{\alpha})=\begin{cases}0, & \eta(\xi;\underline{\alpha})<0\\ \eta(\xi;\underline{\alpha}), & 0\leqslant\eta(\xi;\underline{\alpha})\leqslant 1\\ 1, & \eta(\xi;\underline{\alpha})>1\end{cases} \tag{11}$$

All the biases and synaptic weights are initialized at random, producing a random initial design.

3.1.3. Formulation of variational problem. From Equation (5), the variational statement of this problem is to find a function $\eta^*(\xi;\underline{\alpha}^*)\in V$ for which the integral

$$F[\eta(\xi;\underline{\alpha})]=\int_0^1\eta(\xi;\underline{\alpha})[\eta'(\xi;\underline{\alpha})]^3\,\mathrm{d}\xi \tag{12}$$

defined on V, reaches a minimum value.

Note that no input-target data are used here. Instead and in order to evaluate the objective functional in Equation (12), the integration of a function is needed. For that purpose, Simpson's composite method [30] is applied.

3.1.4. Solution of reduced function optimization problem. The evaluation of the initial guess is 0.565. Training is performed until Brent's method gives zero rate. This occurs when the objective function is 0.4221. Figure 5 illustrates this training process. Note that a logarithmic scale is used for the Y-axis in the gradient norm plot.

Table I shows the training results for this problem. Here N is the number of training epochs, M the number of objective function evaluations, CPU the CPU time for a laptop AMD 3000 in seconds, $\|\underline{\alpha}^*\|$ the final parameters norm, $f(\underline{\alpha}^*)$ the final value for the objective function and $\|\nabla f(\underline{\alpha}^*)\|$ the final gradient norm.

Comparing the drag coefficient provided by that NN (0.4221) to that by the analytical result (0.4220), these two values are almost the same. In particular, the percentage error made by the numerical method is around -0.02%. Finally, the optimal shape by the MLP is shown in Figure 6.

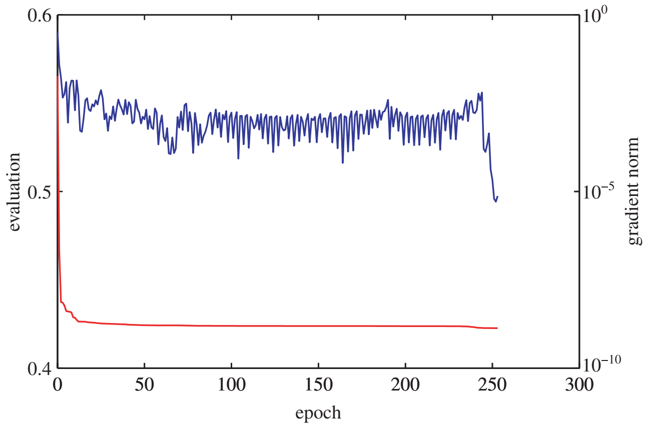

Figure 5. Evaluation (straight line) and gradient norm (zigzag line) training histories for the minimum drag problem.

Table I. Training results for the minimum drag problem.

$N = 253$
$M = 12\,438$
$\text{CPU} = 967$
$\|\alpha\| = 39.288$
$f(\underline{\alpha}^*) = 0.4221$
$\|\nabla f(\underline{\alpha}^*)\| = 7.452 \times 10^{-6}$

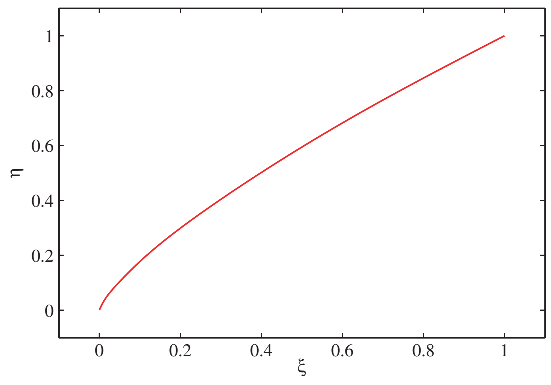

Figure 6. Neural network results to the minimum drag problem.

4. OPTIMAL CONTROL

The objective of optimal control is to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimize or maximize some performance criterion [3]. The formulation of an optimal control problem requires: (i) a mathematical model of the system, that is, a formal description that predicts the response of the physical system to all anticipated inputs; (ii) a statement of the physical constraints on the controls and the states and (iii) a specification of the criterion for evaluating the performance of the system quantitatively.

The performance criterion is a scalar functional of the control variables. In this way, the theory of optimal control is intimately related to the calculus of variations [31]. The history of input values, which makes that objective functional to assume an extremum, is called the optimal control, and the corresponding history of state values is called the optimal trajectory [3].

The application of the variational MLP to the solution of optimal control problems is investigated here through the solution of a minimum time problem, which is a special case of optimal control problem involving independent parameters. In particular, an NN of that type is trained to determine the optimal control and the corresponding optimal trajectory of a car [3].

4.1. Example: the car problem

The car problem is an optimal control problem with two controls, one independent parameter and two state variables. It is defined by an objective functional with two constraints and requiring the simulation of the system dynamics, which is described by two ODEs. The Flood library [29] is also used to solve the problem.

4.1.1. Problem statement. Consider a car that is to be driven along the x-axis from some position x_i at velocity v_i to some desired position x_f at desired velocity v_f in a minimum time t_f.

To simplify the problem, let us approximate the car by a unit point mass that can be accelerated by using the throttle or decelerated by using the brake. Selecting position and velocity as state variables, the mathematical model of this system is given by two ODEs with their corresponding initial conditions:

$$\dot{x}(t) = v(t) \tag{13}$$

$$\dot{v}(t) = a(t) + d(t) \tag{14}$$

$$x(0) = x_i \tag{15}$$

$$v(0) = v_i \tag{16}$$

for $t \in [0, t_f]$ and where the controls $a(t)$ and $d(t)$ are the throttle acceleration and the braking deceleration, respectively.

The acceleration is bounded by the capability of the engine, and the deceleration is limited by the braking system parameters. If the maximum acceleration and deceleration are $\max(a) > 0$ and $\max(d) > 0$, respectively, such bounds on the control variables can be expressed as

$$0 \leqslant a(t) \leqslant \max(a) \tag{17}$$

$$-\max(d) \leqslant d(t) \leqslant 0 \tag{18}$$

Also, the car is to be driven to a desired position x_f and a desired velocity v_f; therefore, $x(t_f)=x_f$ and $v(t_f)=v_f$. Such constraints on the state variables can be expressed as error functionals:

$$\begin{aligned} E_x[(a,d)(t)] &= x(t_f)-x_f \\ &= 0 \end{aligned} \tag{19}$$

$$\begin{aligned} E_v[(a,d)(t)] &= v(t_f)-v_f \\ &= 0 \end{aligned} \tag{20}$$

where E_x and E_v are the final position and velocity errors, respectively.

As the objective is to make the car reach the final point as quickly as possible, the objective functional for this problem is

$$F[(a,d)(t)]=t_f \tag{21}$$

Let us set the initial position, initial velocity, final position, final velocity, maximum acceleration and maximum deceleration to be $x_i=0$, $v_i=0$, $x_f=1$, $v_f=0$, $\max(a)=1$ and $\max(d)=1$, respectively. This optimal control problem has an analytical solution given by

$$a^*(t)=\begin{cases} 1, & 0\leqslant t<1 \\ 0, & 1\leqslant t\leqslant 2 \end{cases} \tag{22}$$

$$d^*(t)=\begin{cases} 0, & 0\leqslant t<1 \\ -1, & 1\leqslant t\leqslant 2 \end{cases} \tag{23}$$

with a minimum final time $t_f^*=2$ [3].

4.1.2. Selection of function space. Here an MLP with a sigmoid hidden layer and a linear output layer is chosen to represent the control $(a,b)(t)$. The NN must have one input, t, and two output neurons, a and d. Again, in this problem we use three hidden neurons. Figure 7 is a graphical representation of this network architecture.

Also this NN needs an associated independent parameter representing the final time t_f. Therefore, such an MLP spans a family V of parameterized functions $(a,b)(t;\underline{\alpha},t_f)$ of dimension $s=14+1$.

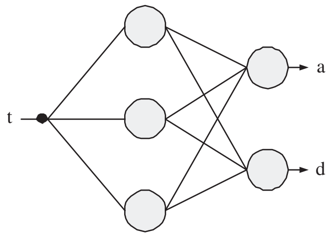

Figure 7. Network architecture for the car problem.

Elements V are of the form

$$\begin{aligned}(a,d):\mathbb{R}&\to\mathbb{R}^2\\ t&\mapsto(a,d)(t;\underline{\alpha},t_f)\end{aligned}$$

where

$$a(t;\underline{\alpha},t_f)=b_1^{(2)}+\sum_{j=1}^{3}w_{1j}^{(2)}\tanh(b_j^{(1)}+w_{j1}^{(1)}t) \tag{24}$$

$$d(t;\underline{\alpha},t_f)=b_2^{(2)}+\sum_{j=1}^{3}w_{2j}^{(2)}\tanh(b_j^{(1)}+w_{j1}^{(1)}t) \tag{25}$$

for $t\in[0,t_f]$. Equation (24) represents just one function, in which many of the parameters are shared.

The control variable is constrained to lie in the interval $[0,1]$. To deal with such constraints, the outputs are bounded in the form

$$a(t;\underline{\alpha},t_f)=\begin{cases}0, & a(t;\underline{\alpha},t_f)<0\\ a(t;\underline{\alpha},t_f), & 0\leqslant a(t;\underline{\alpha},t_f)\leqslant 1\\ 1, & a(t;\underline{\alpha},t_f)>1\end{cases} \tag{26}$$

$$d(t;\underline{\alpha},t_f)=\begin{cases}-1, & d(t;\underline{\alpha},t_f)<-1\\ d(t;\underline{\alpha},t_f), & -1\leqslant d(t;\underline{\alpha},t_f)\leqslant 0\\ 0, & d(t;\underline{\alpha},t_f)>0\end{cases} \tag{27}$$

All the biases and synaptic weights and the independent parameter are initialized here at random. This gives a starting random control with an also randomly chosen final time.

4.1.3. Formulation of variational problem. From Equations (19), (20) and (21), the car problem formulated in this section can be stated so as to find a control $(a,d)^*(t;\underline{\alpha}^*,t_f^*)$ such that

$$E_x[(a,d)^*(t;\underline{\alpha}^*,t_f^*)]=0$$

$$E_v[(a,d)^*(t;\underline{\alpha}^*,t_f^*)]=0$$

and for which the functional

$$T[(a,d)(t;\underline{\alpha},t_f)]=t_f$$

defined on V, takes on a minimum value.

This constrained problem can be converted to an unconstrained one by the use of penalty terms. The statement of this unconstrained problem is now to find a control $[(a,d)^*(t;\underline{\alpha}^*,t_f^*)]$ for which the objective functional

$$F[(a,d)(t;\underline{\alpha},t_f)]=\rho_X(E_x[(a,d)(t;\underline{\alpha},t_f)])^2+\rho_V(E_v[(a,d)(t;\underline{\alpha},t_f)])^2+\rho_T t_f \tag{28}$$

defined on V, takes on a minimum value.

The values $\rho_X=1$, $\rho_V=1$ and $\rho_T=10^{-3}$ are called the error position, error velocity and final time term weights, respectively.

Please note that no input-target data are used here, and that evaluation of the objective functional (28) requires a numerical method for integration of ODEs. Here the Runge–Kutta–Fehlberg method is chosen [30].

4.1.4. Solution of reduced function optimization problem. The evaluation of the initial guess was 0.827; after 112 epochs of training this value falls to 1.999×10^{-3}. Figure 8 shows the complete training history.

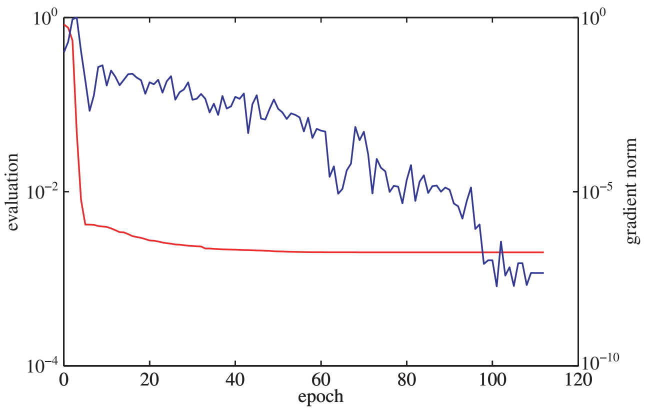

Figure 8. Evaluation (straight line) and gradient norm (zigzag line) training histories for the car problem.

Table II. Training results for the car problem.

$N=112$
$M=7647$
CPU$=324$
$\Vert\underline{\alpha}^*\Vert=6.843$
$f(\underline{\alpha}^*,t_f^*)=1.99951\times10^{-3}$
$e_x(\underline{\alpha}^*,t_f^*)=5.00358\times10^{-4}$
$e_v(\underline{\alpha}^*,t_f^*)=4.99823\times10^{-4}$
$\Vert\nabla f(\underline{\alpha}^*,t_f^*)\Vert=4.63842\times10^{-8}$
$t_f^*=1.99901$

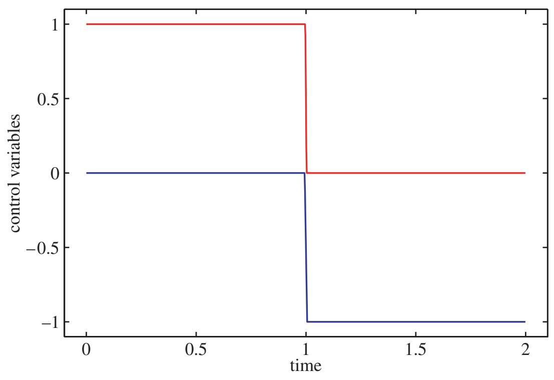

Figure 9. Results for the acceleration (top line) and the deceleration (bottom line) in the car problem.

The training results are summarized in Table II, with N being the number of epochs, M the number of evaluations, CPU the processing time in seconds for a laptop AMD 3000, $\|\alpha\|$ the final parameters norm, f the final objective function value, $\|\nabla f\|$ the final gradient norm, e_x the final position error, e_v the final velocity error and t_f^* the optimum time.

As we can see, the final errors in the position and the velocity of the car are very small, and the final time provided by the NN matches the analytical final time provided by the optimal function in Equation (22). More specifically, the errors made in the constraints are around 10^{-3} and the error made in the final time is about 0.2%. The results for the optimal control signals (acceleration and deceleration) obtained by the NN are shown in Figure 9.

5. INVERSE PROBLEMS

There are two main types of inverse problems: input estimation problems, in which the system properties and output are known and the input is to be estimated; and properties estimation problems, in which the system input and output are known and the properties are to be estimated [4]. The statement of an inverse problem is characterized by (i) a mathematical model of the process depending on the items to be estimated; (ii) a statement of the constraints and (iii) a set of observed data from that process.

Mathematically, inverse problems are a class variational problems. On the other hand, they might be ill-posed [32]. Owing to both their variational and ill-posed nature, inverse problems can be very difficult to solve, and in most cases a direct method is necessary in order to approximate their solution [32].

An artificially generated inverse problem is here attempted for validation. In particular, an MLP is entrusted to estimate the thermal conductivity in an inhomogeneous medium [4].

5.1. *Example: the thermal conductivity estimation problem*

This is an unconstrained problem with two input and one output variables, and where evaluation of the objective functional needs to integrate a PDE. The problem is solved with the Flood library [29]. The PDEs are integrated using the Kratos library [33].

5.1.1. Problem statement. For this problem, consider a two-dimensional inhomogeneous medium with domain Ω and boundary Γ, in which the thermal conductivity is to be estimated. The heat transfer model here is

$$\nabla(\rho(x,y)\nabla T(x,y;t)) = \frac{\partial T(x,y;t)}{\partial t} \quad \text{in } \Omega \tag{29}$$

$$T(x,y;0) = T_0 \quad \text{in } \Omega \tag{30}$$

$$T(x,y;t) = T_\Gamma \quad \text{on } \Gamma \tag{31}$$

for $t \in [0, t_f]$, and where $\rho(x,y)$ is the thermal conductivity, T_0 is the initial temperature and T_Γ is the boundary temperature. Experimental data are obtained from measurements of the temperature for different time steps and at different points on the domain

$$\begin{array}{ccccc}
t_1 & T_{11}(x_1,y_1;t_1) & T_{12}(x_2,y_2;t_1) & \ldots & T_{1Q}(x_Q,y_Q;t_1) \\
t_2 & T_{21}(x_1,y_1;t_2) & T_{22}(x_2,y_2;t_2) & \ldots & T_{2Q}(x_Q,y_Q;t_2) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
t_P & T_{P1}(x_1,y_1;t_P) & T_{P2}(x_2,y_2;t_P) & \ldots & T_{PQ}(x_Q,y_Q;t_P)
\end{array}$$

where P and Q are the number of time steps and points considered, respectively.

The goal in this problem is to estimate the thermal conductivity $\rho(x,y)$; hence, the mathematical model matches the experimental data. In this way, the objective functional for this problem can be the mean-squared error between the computed temperature for a given thermal conductivity and the measured temperature,

$$E[\rho(x,y)] = \frac{1}{PQ}\sum_{i=1}^{P}\left(\sum_{j=1}^{Q}(\hat{T}_{\rho(x,y)}(x_j,y_j;t_i) - T_{ij}(x_j,y_j;t_i))^2\right) \tag{32}$$

For this example, the problem domain is the square $\Omega = \{(x,y) : |x| \leqslant 1, |y| \leqslant 1\}$ with boundary $\Gamma = \{(x,y) : |x| = 1, |y| = 1\}$. Artificial temperature data are generated with the following expression for the thermal conductivity, which will be considered to be the analytical solution of the problem,

$$\rho^*(x,y) = x^2 + y^2 \tag{33}$$

5.1.2. Selection of function space. The thermal conductivity $\rho(x,y)$ in Ω is represented by an MLP with two inputs, three hidden sigmoid neurons a one linear output neuron. Figure 10 is a graphical representation of this network architecture.

Such an NN spans a family V of parameterized functions $\rho(x,y;\underline{\alpha})$ of dimension $s = 13$, which is the number of biases and synaptic weights. Elements of V are of the form

$$\rho : \mathbb{R}^2 \to \mathbb{R}$$

$$(x,y) \mapsto \rho(x,y;\underline{\alpha})$$

where

$$\rho(x,y;\underline{\alpha}) = b_1^{(2)} + \sum_{j=1}^{3} w_{1j}^{(2)} \cdot \tanh(b_j^{(1)} + w_{j1}^{(1)}x + w_{j2}^{(1)}y) \tag{34}$$

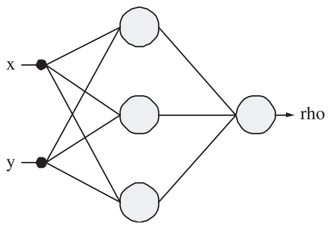

Figure 10. Network architecture for the thermal conductivity estimation problem.

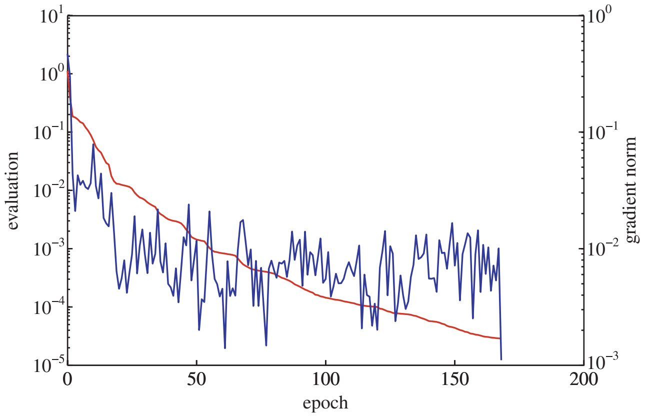

Figure 11. Evaluation (straight line) and gradient norm (zigzag line) training histories for the thermal conductivity estimation problem.

On the other hand, the biases and synaptic weights are initialized at random, which means that a random initial guess is used for the thermal conductivity.

5.1.3. Formulation of variational problem. Making use of Equation (32), the variational statement of this problem is to find a function $\rho^*(x, y; \underline{\alpha}^*) \in V$ for which the functional

$$E[\rho(x, y; \underline{\alpha})] = \frac{1}{PQ} \sum_{i=1}^{P} \left(\sum_{j=1}^{Q} (\hat{u}_{\rho(x,y;\underline{\alpha})}(x_j, y_j; t_i) - u_{ij}(x_j, y_j; t_i))^2 \right) \tag{35}$$

defined on V, achieves a minimum.

Evaluation of the objective functional (35) requires a numerical method for integrating PDEs. Here the finite element method [34] is chosen. The domain is discretized with a triangular mesh composed of 927 elements and 568 nodes, and the time is discretized in 11 time steps.

5.1.4. Solution of reduced function optimization problem. The evaluation of the initial guess is 1.117. Convergence occurs after 168 epochs. After training the evaluation falls to 2.868×10^{-5}. The gradient norm decreases from 0.469 to 1.106×10^{-3}. Figure 11 depicts the evaluation value of the objective function and its gradient norm *versus* the training epoch.

Table III. Training results for the thermal conductivity estimation problem.

$N = 168$
$M = 10\,476$
$\text{CPU} = 16\,253$
$\|\underline{\alpha}^*\| = 21.606$
$f(\underline{\alpha}^*) = 2.868 \times 10^{-5}$
$\|\nabla f(\underline{\alpha}^*)\| = 1.106 \times 10^{-3}$

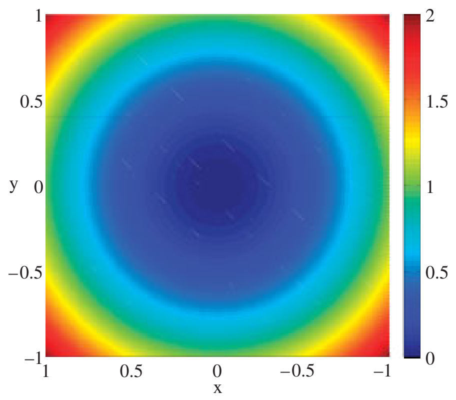

Figure 12. Neural network results to the thermal conductivity estimation problem.

Table III shows the training results for this problem. Here N denotes the number of training epochs, M the number of objective functional evaluations, CPU the CPU time in seconds for a laptop AMD 3000, $\|\alpha\|$ the final parameters norm, $f(\underline{\alpha}^*)$ the final objective function evaluation and $\|\nabla f(\underline{\alpha}^*)\|$ its gradient norm.

The solution here is good, since the mean-squared error between the output from the model and the experimental data is of order 10^{-6} and the estimated thermal conductivity matches very well the actual thermal conductivity given by Equation (33). Figure 12 shows the thermal conductivity estimated by the NN.

6. CONCLUSIONS AND FUTURE WORK

A variational formulation for NNs provides a direct method for the solution of variational problems. The performance of this numerical method has been studied through different classes of engineering applications, such as shape design, optimal control and inverse analysis.

The examples considered were efficiently solved by the proposed methodology. It is important to highlight that the results, practically equivalent to the analytical ones, where achieved with very few degrees of freedom. An extra advantage of the proposed technique is that, at least for these problems, no suboptimal solutions were found. This might indicate that, in contrast to other direct methods, multimodality is significantly reduced. This aspect needs further exploration and it is subject of current research.

REFERENCES

1. Gelfand IM, Fomin SV. *Calculus of Variations*. Prentice-Hall: Englewood Cliffs, NJ, 1963.
2. Bucur D, Buttazzo G. *Variational Methods in Shape Optimization Problems*. Birkhauser: Basel, 2005.
3. Kirk DE. *Optimal Control Theory. An Introduction*. Prentice-Hall: Englewood Cliffs, NJ, 1970.
4. Kirsch A. *An Introduction to the Mathematical Theory of Inverse Problems*. Springer: Berlin, 1996.
5. Eyi S, Hager JO, Lee KD. Airfoil design optimization using the Navier–Stokes equations. *Journal of Optimization Theory and Applications* 1994; **83**(3):447–461.
6. Park S, Ramirez WF. Optimal production of secreted protein in fed-bath reactors. *AIChE Journal* 1988; **34**(9):1550.
7. Yin H, Yao M. Inverse problem-based analysis on non-uniform profiles of thermal resistance between strand and mould for continuous round billets casting. *Journal of Materials Processing Technology* 2007; **183**(1):49–56.
8. Elsgolc LE. *Calculus of Variations*. Pergamon Press: Oxford, 1961.
9. Hwang C, Shih YP. Laguerre series direct method for variational problems. *Journal of Optimization Theory and Applications* 1983; **39**(1):143–149.
10. Chang RY, Wang ML. Shifted Legendre direct method for variational problems. *Journal of Optimization Theory and Applications* 1983; **39**(2):299–307.
11. Horng IR, Chou JH. Shifted Chebyshev direct method for solving variational problems. *International Journal of Systems Science* 1985; **16**(77):855–861.
12. Hsiao CH. Solution of variational problems via Haar orthonormal wavelet direct method. *International Journal of Computer Mathematics* 2004; **81**(7):871–887.
13. Haykin S. *Neural Networks*: *A Comprehensive Foundation*. Prentice-Hall: Englewood Cliffs, NJ, 1994.
14. Bishop C. *Neural Networks for Pattern Recognition*. Oxford University Press: Oxford, 1995.
15. Lecoeuche S, Tsaptsinos D. Presenting the special issue on Engineering Applications of Neural Networks—novel applications of neural networks in engineering. *Engineering Applications of Artificial Intelligence* 2006; **19**(7):719–720.
16. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Networks* 1989; **2**(5):359–366.
17. Zoppoli R, Sanguineti M, Parisini T. Approximating networks and extended Ritz method for the solution of functional optimization problems. *Journal of Optimization Theory and Applications* 2002; **112**(2):403–439.
18. Sarkar D, Modak JM. ANNSA: a hybrid neural network/simulated annealing algorithm for optimal control problems. *Chemical Engineering Science* 2003; **58**:3131–3142.
19. Franco-Lara E, Weuster-Botz D. Estimation of optimal feeding strategies for fed-bach bioprocesses. *Bioprocess and Biosystems Engineering* 2005; **27**(4):255–262.
20. Šíma J, Orponen P. General-purpose computation with neural networks: a survey of complexity theoretic results. *Neural Computation* 2003; **15**:2727–2778.
21. Demuth H, Beale M. *Neural Network Toolbox for Use with MATLAB. User's Guide*. The MathWorks: Natick, MA, U.S.A., 2002.
22. Lopez R, Oñate E. A variational formulation for the multilayer perceptron. *Proceedings of the 16th International Conference on Artificial Neural Networks ICANN 2006*, Athens, Greece, 2006.
23. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in C++*: *the Art of Scientific Computing*. Cambridge University Press: Cambridge, 2002.
24. Fogel DB. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks* 1994; **5**(1):3–14.
25. Kennedy J, Eberhart RC. Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks*, Perth, WA, Australia, 1995; 1942–1948.
26. Pardalos PM, Romeijna HE, Tuyb H. Recent developments and trends in global optimization. *Journal of Computational and Applied Mathematics* 2000; **124**:209–228.
27. Mohammadi B, Pironneau O. Shape optimization in fluid mechanics. *Annual Review of Fluid Mechanics* 2004; **36**:255–279.
28. Brown SL, Hull DG. Axisymmetric bodies of minimum drag in hypersonic flow. *Journal of Optimization Theory and Applications* 1969; **3**(1):52–71.

29. Lopez R. *Flood*: *An Open Source Neural Networks C++ Library*. www.cimne.com/flood, 2006.
30. Stoer J, Bulirsch R. *Introduction to Numerical Analysis*. Springer: Berlin, 1980.
31. Betts JT. A survey of numerical methods for trajectory optimization. *AIAA Journal of Guidance*, *Control and Dynamics* 1998; **21**(2):193–207.
32. Sabatier PC. Past and future of inverse problems. *Journal of Mathematical Physics* 2000; **41**:4082–4124.
33. Dadvand P. *Kratos*: *An Object-Oriented Environment for Development of Multi-Physics Analysis Software*. www.cimne.upc.edu/kratos, 2006.
34. Zienkiewicz OC, Taylor RL, Zhu JZ. *The Finite Element Method*: *ITS Basis and Fundamentals*. Elsevier: Amsterdam, 2005.