Neural Network Differential Equation and Plasma Equilibrium Solver

B. Ph. van Milligen, V. Tribaldos, and J. A. Jiménez
Asociación EURATOM-CIEMAT para Fusión, Avenida Complutense 22, 28040 Madrid, Spain
(Received 10 March 1995)

A new generally applicable method to solve differential equations, based on neural networks, is proposed. Straightforward to implement, finite differences and coordinate transformations are not used. The neural network provides a flexible and compact base for representing the solution, found through the global minimization of an error functional. As a proof of principle, a two-dimensional ideal magnetohydrodynamic plasma equilibrium is solved. Since no particular topology is assumed, the technique is especially promising for the three-dimensional plasma equilibrium problem.

PACS numbers: 07.05.Mh, 02.60.Lj, 52.55.Fa, 52.55.Hc

In this Letter we propose a new method to solve differential equations based on neural networks. To demonstrate the potential of the method, it is applied to the two-dimensional plasma equilibrium problem.

The general problem to which the method is applicable is of the form

$$D(\vec{f}(\vec{x})) = \vec{0}, \tag{1}$$

where D is a differential operator (which may be nonlinear and nonhomogeneous) and $\vec{f}$ is a mulitvariate function of $\vec{x}$ that satisfies appropriate boundary conditions so that it is a unique solution to Eq. (1).

The new method consists in *approximating* the solution $\vec{f}(\vec{x})$ with a neural network. The training process of the neural network is a little different from the usual procedure (which consists in training the network by presenting it with examples of a *known* solution): Here we adjust the network weights by means of a variant of the error backpropagation algorithm in order to minimize (a) Eq. (1) on the whole domain within the given boundaries and (b) a penalty functional that depends on the boundary conditions.

We approximate the solution using a network of the multilayer perceptron type with one hidden layer (MLP-1). This type of network is capable of approximating an arbitrary continuous function to within arbitrary precision, provided sufficient hidden nodes are present [1–6].

Denoting the network inputs by $\vec{x} = \{x_1, x_2, \ldots, x_J\}$, the hidden layer nodes by $\vec{y} = \{y_1, y_2, \ldots, y_K\}$, and the network outputs by $\vec{z} = \{z_1, z_2, \ldots, z_L\}$, we define

$$y_k = \sigma\left[\sum_{j=1}^{J+1} v_{kj} x_j\right] \quad \text{and} \quad z_l = \sum_{k=1}^{K+1} w_{lk} y_k, \tag{2}$$

where x_{J+1} and y_{K+1} are so-called "bias units" with a fixed value of 1, v, and w are referred to as input and output weights, respectively, and $\sigma(x)$ is the sigmoid function:

$$(x) = \tanh(x/2). \tag{3}$$

We note that it is usually beneficial to perform a linear scaling of the input ($\vec{x}$) and output ($\vec{z}$) variables such that they are of order 1.

We intend that the network output $\vec{z}$ should provide an approximation to the exact solution of Eq. (1), $\vec{f}(\vec{x})$. For that purpose, we define a penalty function E:

$$E = [E_{\mathrm{de}}]^2 + [E_{\mathrm{bc}}]^2, \tag{4}$$

where $E_{\mathrm{de}} \equiv D(\vec{z})$ is the left-hand side of the differential equation Eq. (1) applied to $\vec{z}$, and E_{bc} schematically represents a functional that is zero if and only if the boundary conditions are satisfied. For $\vec{z} = \vec{f}$, $E = 0$, and since the solution is unique, minimizing Eq. (4) provides an approximation to $\vec{f}$.

Note that due to the simplicity of the network specified by Eq. (2), it is generally possible to give an analytic expression for the evaluation of the differential operator D for a given set of network weights, making the implementation of the method straightforward for a large class of differential equations.

$E_{\mathrm{de}}(\vec{x})$ is evaluated for a (sufficiently large) number N of values $\{\vec{x}(i), i = 1, \ldots, N\} \in \Omega$, where Ω is a finite domain in $\mathbb{R}^J$; similarly, $E_{\mathrm{bc}}(\vec{x})$ is evaluated for a number N_{bound} of values $\{\vec{x}(j), j = 1, \ldots, N_{\mathrm{bound}}\} \in \partial\Omega$. The *total* error E_{tot} is defined as the sum of squares all the values of $E_{\mathrm{de}}(x(i))$ and $E_{\mathrm{bc}}(x(j))$ (with appropriate weights). Minimization of E_{tot} will then provide a solution to Eq. (1) on Ω, satisfying the boundary conditions as specified, provided (a) the number of hidden nodes is sufficient, (b) the covering of the domain and its boundaries by $\{\vec{x}(i)\}$ is well distributed so that the solution is well described by the function values $\{f(\vec{x}(i))\}$, and (c) this covering is dense enough to avoid overfitting ($N + N_{\mathrm{bound}}$ is larger than the number of free parameters or network weights). The number of hidden nodes then determines the maximum attainable accuracy.

For the minimization procedure we use a standard quasi Newton gradient-descent algorithm. The gradients of E [Eq. (4)] with respect to the weights v and w can be expressed analytically if the problem is formulated properly. The formulation of the solution method as given above is sufficiently general to permit its direct application to a wide range of problems in science.

In the following, we apply this method to two test cases. Both are elliptic second-order partial differential

 0031-9007/95/75(20)/3594(4)$06.00

equations that appear in the framework of the ideal magnetohydrodynamic (MHD) plasma equilibrium problem.

An analytic high-β tokamak equilibrium.—This example has most of the ingredients of a full two-dimensional equilibrium problem. It is stated as follows:

$$E_{\rm de} = \nabla^2\psi - A - Cr\cos(\theta) = 0, \quad (5)$$

where ψ is the poloidal flux function, A and C are constants related to the profiles of the toroidal field and the pressure, respectively, and (r, ϕ, θ) defines a quasicylindrical coordinate system related to the usual cylindrical coordinate system (R, ϕ, Z) by $R = R_0 + r\cos(\theta)$ and $Z = r\sin(\theta)$, where $R_0 = 0.6$ m is the tokamak major radius (here ϕ is the ignorable coordinate). When the boundary condition $E_{\rm bc} = \psi(r = a) = 0$ is imposed ($a = 0.1$ m being the plasma minor radius), the exact solution is [7]

$$\psi_{\rm analytic} = \tfrac{1}{8}(r^2 - a^2)[2A + Cr\cos(\theta)]. \quad (6)$$

We have selected $A = 1$ and $C = 10$ (giving a cylindrical safety factor of $q* = 2$ and an average plasma pressure, normalized to the toroidal magnetic field pressure, of $\beta_t = 0.21$), and made the identifications $x_1 = R - R_0$, $x_2 = Z$ (two input nodes) and $\alpha z_1 = \psi$ (one output node; here $\alpha = 10^{-3}$ is a constant scaling factor used only to obtain values of z_1 of order of 1). Using the method outlined above, the cost function $E = E_{\rm de}^2 + E_{\rm bc}^2$ is then minimized on an appropriate poloidal grid with a small neural network having only $K = 15$ hidden nodes. Convergence of the minimization, starting with random network weights, is extremely rapid: In about 500 iterations a minimum is found with an average error of 0.12% in ψ. Figure 1 shows the level contours of $\psi_{\rm net}$ as obtained from the trained network [visually indistinguishable from the analytic solution, Eq. (6)]. Figure 2 shows the reconstruction error (i.e., $\psi_{\rm net} - \psi_{\rm analytic}$) vs $\psi_{\rm analytic}$ for 2000 points on an equally spaced (r, θ) grid.

A Grad-Shafranov solver for fixed-boundary tokamak equilibria.—The solution of the tokamak plasma equilibrium problem is well studied [7]. The problem can be stated as follows: determine the poloidal flux function $\psi(R, Z)$ such that the Grad-Shafranov equation is satisfied:

$$E_1^2 = \sum_{n=1}^{N}\left[\Delta^*\psi + \mu_0 R^2\frac{\partial p}{\partial\psi} + F\frac{\partial F}{\partial\psi}\right]_\Omega^2 = 0, \quad (7)$$

where the Grad-Shafranov operator is $\Delta^*\psi = R\partial[(1/R)\partial\psi/\partial R]/\partial R + \partial^2\psi/\partial Z^2$, $p(\psi)$ is the pressure profile, and $F(\psi)$ is related to the poloidal current profile. Equation (7) is evaluated for N interior points of the plasma cross section (Ω). This equation, along with adequate boundary conditions and a suitable specification of the two source profiles $p(\psi)$ and $F(\psi)$ describes toroidally symmetric ideal MHD equilibria. To eliminate the arbitrary integration constant in ψ, and to give to ψ its physical meaning of a magnetic field integrated over a surface, we require that there is a magnetic axis $(R_{\rm ax}, Z_{\rm ax})$ where the flux ψ attains its minimum, being equal to 0:

$$E_2^2 = [\psi(R_{\rm ax}, Z_{\rm ax}) - 0]^2 = 0. \quad (8)$$

The location of the magnetic axis $(R_{\rm ax}, Z_{\rm ax})$ is not known *a priori*. Below we will explain how we treat this problem.

A suitable boundary condition for the differential equation is $\vec{B}\cdot\vec{n} = 0$ at the plasma boundary (fixed-boundary problem with a perfectly conducting wall), where $\vec{B}$ is the magnetic field and $\vec{n}$ is the normal to the boundary, or

$$E_3^2 = \sum_{m=1}^{N_{\rm bound}}[\psi - \psi_{\rm bound}]_{\partial\Omega}^2 = 0, \quad (9)$$

which is evaluated for $N_{\rm bound}$ boundary points (on $\partial\Omega$). The penalty functional $E_{\rm tot}$ for minimization then becomes $E_{\rm tot} = \sum_{i=1}^{3}\alpha_i\gamma_i E_i^2$, where α_i and γ_i are weight factors. The factors γ_i are chosen equal to the inverse of the desired values of each E_i^2, $\gamma_i = (E_i^{(0)})^{-2}$. Thus, for a satisfactory solution, $\gamma_i E_i^2 \approx 1$. The factors α_i are initially equal to 1 and are adapted as the minimization algorithm advances: every $N_{\rm eval}$ iteration the α_i are reset to the values $\alpha_i = \max(\frac{1}{4}, \gamma_i, E_i^2/\frac{1}{3}\sum_{i=1}^{3}\gamma_i E_i^2)$, such that more work is invested in the E_i that is worse. Every $N_{\rm eval}$ iteration also the minimum of ψ as given by the partially entrained network is determined, and the values of

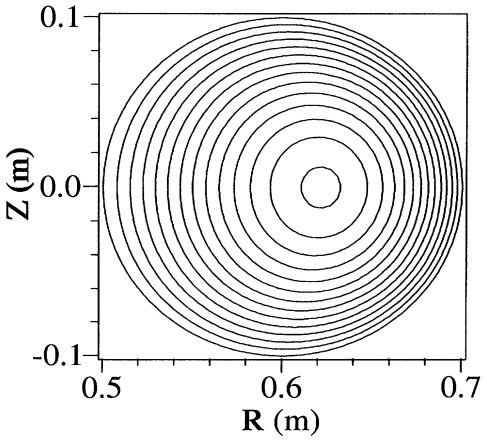

FIG. 1. Contours of $\psi_{\rm net}$, as computed from the neural network solution, for the analytic high-β case.

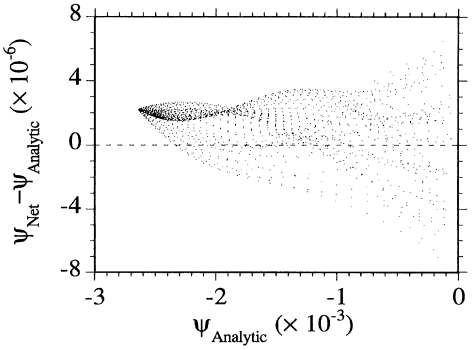

FIG. 2. Error of the neural network flux $\psi_{\rm net}$ with respect to the analytical flux $\psi_{\rm analytic}$ from the exact solution for the high-β case.

(R_{ax}, Z_{ax}) are changed to indicate this position. Thus the magnetic axis is updated every N_{eval} iteration and gradually approximates its true position.

The N interior points of Ω and the N_{bound} points on the boundary $\partial\Omega$ must be chosen to cover both regions homogeneously. Should any large "gaps" be present, then the solution produced by the network in those regions cannot be trusted. The total number of data points is $N + 1 + N_{bound}$. This number should be large enough to avoid overfitting. The total number of "free parameters" of the minimization problem is given by the number of network weights, or $4K + 1$ [cf. Eq. (2) with $J = 2$ and $L = 1$]. Thus overfitting can be avoided when $N + 1 + N_{bound} \gg 4K + 1$.

We choose the same representation of the flux by the neural network as in the previous example, except that here $\alpha = 1$. All gradients needed for the gradient-descent minimization algorithm (derivatives of ψ with respect to R and Z are derivatives of E_{tot} with respect to the network weights) are evaluated *analytically*.

To check the solution we have compared it with the solution generated by the plasma equilibrium solver VMEC [8–10] for a particular case. VMEC calculates an equilibrium, assuming nested flux surfaces, from input profiles $p(\psi)$ and $q(\psi)$, where the safety factor q is related to the current distribution. The equilibrium problem expressed by Eqs. (7)–(9) is slightly different, since it requires the profiles $p(\psi)$ and $F(\psi)$ as input. To obtain the same equilibrium with both methods, we have passed the VMEC output profile $F(\psi)$ on to the neural network solver along with the input profile $p(\psi)$. Both profiles are given in terms of polynomial fit coefficients to the VMEC profiles.

We have selected a D-shaped plasma (JET-like [11]) with $R_0 = 3.1$ m, horizontal minor radius $a = 1.35$ m, elongation $\kappa = 1.66$, a safety factor at the boundary of $q_a = 4$, and a total average normalized pressure of $\langle\beta\rangle = 4.26\%$.

In order to have an objective criterion for the quality of the equilibrium obtained from this procedure, we introduce the force balance error $|\varepsilon| = |\vec{j} \times \vec{B} - \vec{\nabla} p|/|\vec{\nabla} p|$, where $\mu_0 \vec{j} = (\vec{\nabla} F \times \vec{e}_\phi - \Delta^* \psi \cdot \vec{e}_\phi)/R$ and $\vec{B} = (F \cdot \vec{e}_\phi - \vec{\nabla}\psi \times \vec{e}_\phi)/R$. These quantities can be evaluated directly from the solution $\psi(R, Z)$.

Accuracy of the solution.—Using a network with $K = 31$ hidden nodes, the neural network solver converges in 1000 iterations to a solution with an average force balance error of $\langle|\varepsilon|\rangle = 2.63\%$. Figure 3 shows the flux contours as obtained from the network solver and the location of the corresponding VMEC flux surfaces. The average flux error is $\langle\Delta\psi^2\rangle^{1/2} = 1.82 \times 10^{-3}$ $(0 \le \psi \le 0.304)$. A more accurate reconstruction can be obtained by increasing the number of iterations or the number of hidden nodes. Figure 4 compares the force balance error, averaged over the flux surfaces, for the VMEC equilibrium and the network solution with $K = 31$ and 127 nodes ($\langle|\varepsilon|\rangle = 1.22\%$, $\langle\Delta\psi^2\rangle^{1/2} = 2.33 \times 10^{-3}$). The VMEC error increases near the magnetic axis and the boundary

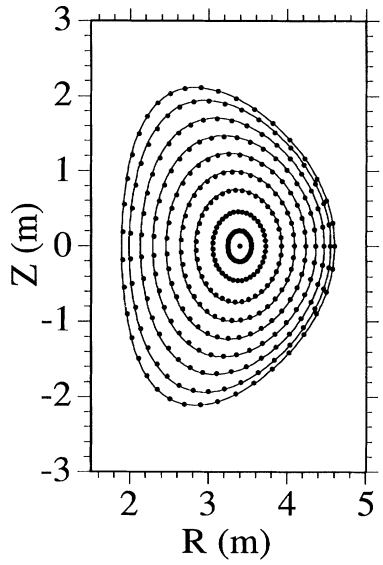

FIG. 3. Flux surfaces as generated by the VMEC equilibrium solver and the neural network solver. Dots indicate the location of the VMEC flux surfaces; lines are contour levels of the neural network output at the same flux values. The correspondence is quite good, especially considering the small number of hidden nodes of the network ($K = 31$).

since it solves the equation in flux coordinates, so that these points are singularities. The distribution of the network error is smooth. It should be noted that whereas VMEC minimizes the errors on each flux surface separately, the network minimizes the errors globally. Figure 5 shows the q profile for the $K = 127$ case as compared to the VMEC profile where

$$q = \frac{1}{2\pi} \oint_{\psi=\text{const}} \frac{B_\phi}{RB_\theta}\, ds = \frac{F(\psi)}{2\pi} \oint_{\psi=\text{const}} \frac{ds}{R^2 B_\theta}.$$

Note that both the flux surface reconstruction (Fig. 3) and the q-profile reconstruction (Fig. 5) are satisfactory, even if the force balance error is larger than the VMEC error (Fig. 4).

Speed of the calculations.—In the above calculations, fifth-order polynomial fits were used to represent the

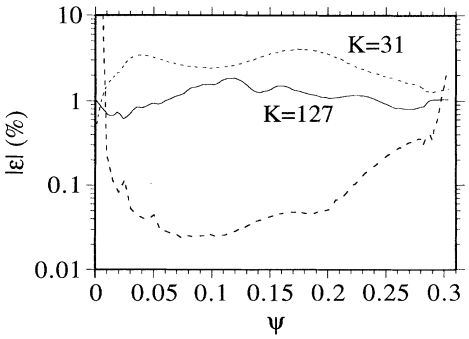

FIG. 4. Force balance error for the VMEC equilibrium (dashed line) and the network equilibrium with both $K = 31$ and 127.

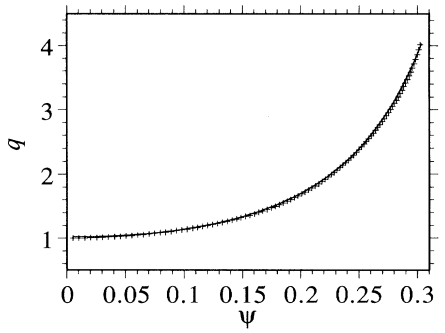

FIG. 5. Profiles of the safety factor q, as calculated by VMEC (crosses) and the neural network solver (solid line). The difference is negligible, even though the difference in the force balance error is considerable (cf. Fig. 4).

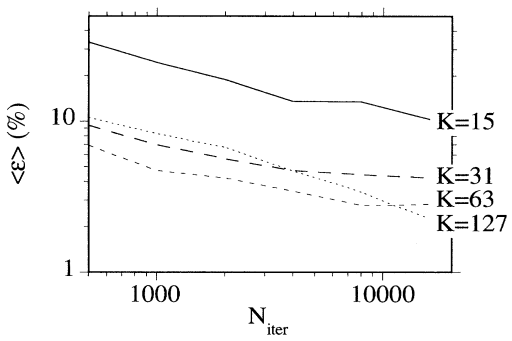

FIG. 6. Dependence of the volume-average force balance error on the number of hidden nodes of the network (K) and the number of iterations.

profiles $F(\psi)$ and $p(\psi)$ to obtain the same equilibrium as VMEC to high accuracy. For high speed calculations, we use third-order polynomials. This does not reduce the validity of the obtained equilibrium in any way, since the original profiles were slightly arbitrary. Figure 6 shows the way in which the volume-average force balance error drops with K and the number of iterations. The CPU time for the calculation is approximately given by $t_{\mathrm{CPU}} = \gamma N_{\mathrm{ITER}}(N + 1 + N_{\mathrm{bound}})(4K + 1)$, where γ is a proportionality constant that depends on the computer used. On our Cray YMP-EL (about 5 times slower than a Cray YMP), $\gamma = 1.87\ \mu$s. The solution with $K = 31$ nodes and $N_{\mathrm{ITER}} = 1000$ is obtained in a time comparable to the VMEC solution (134 s).

Prospects.—Encouraged by these results, the calculation of fully three-dimensional (stellarator) equilibria will be undertaken in the near future. An excellent representation of the flux of a Heliac stellarator equilibrium (TJ-II) has already been obtained with only $K = 255$ hidden nodes [12], suggesting that the three-dimensional equilibrium problem is tractable by this method. Whereas in the solution of the two-dimensional tokamak equilibrium the VMEC code and the network use a similar number of free parameters, this number is about ten times larger for the three-dimensional VMEC equilibrium TJ-II than for the network. Thus the network provides a more compact representation which may be advantageous. Further, the fact that no assumptions need to be made with respect to the topology of the solution (nested flux surfaces) is of special interest for the three-dimensional case. Finally, the method promises to be fast for problems that have to be solved repeatedly with similar boundary conditions, because the solver can use the previous solution as its starting point.

In conclusion, the present relatively simple cases provide a proof of principle of the power and possibilities of the new solution method. We stress the major advantages of the method: (1) It is straightforward to implement for a wide class of problems. (2) Finite differences are not used in the solution process. (3) The equations are solved in real space, i.e., without complicated and costly coordinate transformations. (4) For the ideal MHD equilibrium problem, no particular topology of the solution (nested flux surfaces) is supposed. Thus the possibility of incorporating X points or islands exists, insofar as permitted by the equations. Finally, we point out that the method seems quite generally applicable to the solution of multivariate differential equations with boundary conditions on a finite domain that possess a unique solution. We feel, however, that a rigorous mathematical treatment of the stability and convergence properties of this method would be welcome.

This work was made possible by Commission of the European Communities Bursary ERB4001GT921485. We would like to thank our colleagues at the Asociación EURATOM-CIEMAT for a very pleasant cooperation.

[1] P. T. Wasserman, *Advanced Methods in Neural Computing* (Van Nostrand Reinhold, New York, 1993).
[2] K. Hornick, M. Stinchombe, and H. White, Neural Networks **2**, 359 (1989).
[3] K. Hornick, Neural Networks **4**, 251 (1991).
[4] K. Funahashi, Neural Networks **2**(3), 183 (1989).
[5] Tarun Khanna, *Foundations of Neural Networks* (Addison-Wesley, Reading, MA, 1990).
[6] L. Allen and C. M. Bishop, Plasma Phys. Controlled Fusion **34**, 1291 (1992).
[7] J. P. Freidberg, *Ideal Magnetohydrodynamics* (Plenum Press, New York, 1987) (Chap. 6, and references therein).
[8] S. P. Hirshman and J. C. Whitson, Phys. Fluids **26**, 3553 (1983).
[9] S. P. Hirshman and D. K. Lee, Comput. Phys. Commun. **39**, 161 (1986).
[10] S. P. Hirshman and O. Betancourt, J. Comput. Phys. **96**, 99 (1990).
[11] The JET Project: Design Proposal of the Joint European Torus, EUR-5516e (EUR-JET-R5), Commission of the European Communities.
[12] V. Tribaldos and B. Ph. van Milligen, Nucl. Fusion (to be published).