2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Feedback Neurocontrol of a Disease

Danil V. Prokhorov

***Abstract*— We consider a simple mathematical model of a disease. It includes four coupled nonlinear equations and four state variables for pathogens, plasma cells, antibodies and patient health indicator. Depending on initial values of state variables (concentrations) and without control, the model exhibits at least four possible classes of behavior. We treat the system of equations from the standpoint of feedback neurocontrol. We show that various feedback control strategies (due to injection of therapeutic agents) are possible and effective. We also discuss ways of making the "patient controller" more robust with respect to realistic uncertainties.**

I. Introduction

In an organism, a disease is often represented as a dynamic process. Such a process usually begins with the growth of pathogenic agents in strength and quantity. Depending on the power of its opposition (natural antibodies and/or medical treatment), the disease may result in the following four outcomes. In subclinical and clinical cases, the pathogen is destroyed, and the affected organ recovers fully, although it takes longer for the recovery in the clinical case. In the chronic case, the organ receives some permanent damage. If the pathogen is too strong, or alternatively the natural or artificial defenses are insufficiently powerful, then the pathogen causes death of the organ.

Though there exists a body of literature about control of drug delivery and healthcare process, etc. (e.g., [1], [2], [3], [4]), it is interesting to consider unconventional methods of controlling diseases. Here we study application of neural networks as "patient controller", i.e., the system which uses quantities that can conceivably be measured in real time and processes them to deliver the appropriate level of therapy. Our model contains realistic but not specific features of disease dynamics including an organ, an immune system and a pathogen. We assume that therapeutic agents are idealized in its influence on the disease. We also consider parametric uncertainty of the disease model, as well as measurement noise, which together add realism to our study.

We first describe the disease model (Section II), then discuss the cost function to be minimized and our approach (Section III). In Section IV we illustrate application of our approach with several examples. All examples are representative of various feedback control strategies to treat the disease.

Danil V. Prokhorov is with Toyota Technical Center, Ann Arbor, MI 48105, USA (email: dvprokhorov@gmail.com).

II. Disease Dynamic Model

A simple disease dynamic model is introduced in [5]:

$$\begin{aligned}\dot{x}_1 &= (a_{11} - a_{12}x_3)x_1 + b_1u_1\\ \dot{x}_2 &= a_{21}(x_4)a_{22}x_1x_3 - a_{23}(x_2 - x_2^*) + b_2u_2\\ \dot{x}_3 &= a_{31}x_2 - (a_{32} + a_{33}x_1)x_3 + b_3u_3\\ \dot{x}_4 &= a_{41}x_1 - a_{42}x_4 + b_4u_4\end{aligned} \quad (1)$$

where x_1 is concentration of a pathogen, x_2 is concentration of plasma cells, x_3 is concentration of antibodies (they destroy the pathogen), and x_4 is organ damage indicator. While the original model in [5] is uncontrolled, we add four control variables which correspond to application of therapeutic agents. Thus, u_1 is a pathogen killer, u_2 is a plasma cell enhancer, u_3 is an antibody enhancer, and u_4 is a health enhancer which helps the organ to heal.

The "nominal" values of a_{ij} and b_{ij} are $a_{11} = 1.0$, $a_{12} = 1.0$, $a_{22} = 3.0$, $a_{23} = 1.0$, $a_{31} = 1.0$, $a_{32} = 1.5$, $a_{33} = 0.5$, $a_{41} = 0.5$, $a_{42} = 1.0$, $b_1 = b_4 = -1.0$ and $b_2 = b_3 = 1.0$ (see [6]). The parameter a_{21} is the nonlinear function

$$a_{21}(x_4) = \begin{cases}0, & \text{if } x_4 \geq 0.5;\\ \cos(\pi x_4), & \text{if } 0 \leq x_4 < 0.5.\end{cases} \quad (2)$$

The state variables x_i and controls u_j can be assembled into the state $\mathbf{x}$ and control $\mathbf{u}$ vectors, respectively. It should be noted that both the state and the control vectors are nonnegative. We denote $x_4 = 0$ as the healthy organ state, and $x_4 \geq 1.0$ as the organ death.

The equations (1) are coupled, with the most involved equation being that of plasma cell dynamics (x_2). Given the initial concentration of the pathogen x_1, the uncontrolled disease model ($u_j = 0$, $\forall j$) can exhibit up to four cases of disease progression mentioned in Introduction. The initial populations of the plasma cells and the antibodies are assumed to be constant. When $x_1(0) \approx 1.5$ we have the subclinical case, requiring no therapeutic agent intervention. The clinical case occurs when $x_1(0) \approx 2.0$. The organ can still fully recover on its own. When the pathogen concentration increases to ≈ 2.6 the organ sustains some permanent damage, while the production of antibodies is suppressed, and the pathogen remains present in the body. Finally, if $x_1(0) \approx 3.0$ the organ will die unless therapeutic agents are applied effectively. These four cases are illustrated in Figure 1 for $\mathbf{u} = \mathbf{0}$ over a reasonable period of time.

III. Our approach

In order to apply our previously developed gradient-based methodology of neuroconrollers [7], [8] we first discretize (1). For simple yet sufficiently accurate implementation, we

0-7803-9490-9/06/$20.00/©2006 IEEE

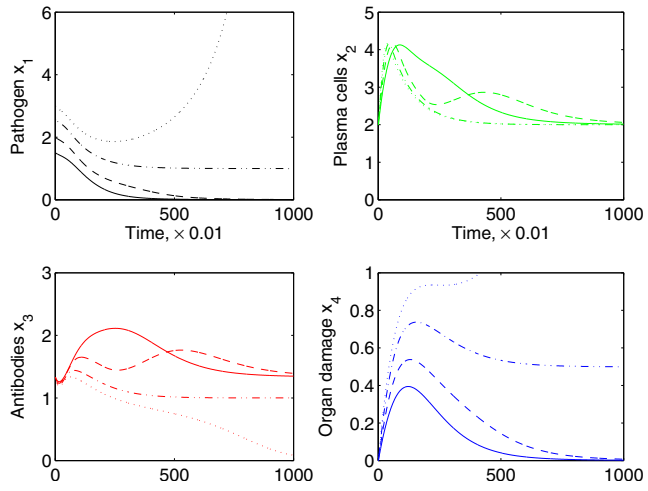

Fig. 1. Disease progress when therapy is absent. The organ recovery depends on the initial concentration of the pathogen x_1. The subclinical and clinical cases (solid and dashed lines, respectively) require no medical intervention. The chronic case (dash-dotted line) may result in a permanent damage of the organ, whereas the lethal case (dotted line) ends up in its death.

use the first-order Euler method with the fixed step size of 0.01. We can then write the system in the general discrete-time form

$$\begin{aligned} \mathbf{u}_k &= \mathbf{g}(\mathbf{b}, \mathbf{x}_k, \mathbf{w}) \\ \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{a}, \mathbf{x}_k, \mathbf{u}_k) \end{aligned} \tag{3}$$

where $\mathbf{w}$ is the vector of neural network weights.

Our performance measure to be minimized is

$$J = \langle \sum_{k=1}^{N} U(\mathbf{x}_k, \mathbf{u}_k) \rangle \tag{4}$$

$$\begin{aligned} U(\mathbf{x}_k, \mathbf{u}_k) = {} & \big(q_1 x_{k,1}^2 + q_4 x_{k,4}^2 + \\ & r_1 u_{k,1}^2 + r_2 u_{k,2}^2 + r_3 u_{k,3}^2 + r_4 u_{k,4}^2\big)\Delta t \end{aligned} \tag{5}$$

where $< \cdot >$ denotes expectation with respect to $\mathbf{x}$ and, possibly, $\mathbf{a}$ and $\mathbf{b}$ if they change with time; q_i and r_j are positive weighting factors. The value of N is not defined a priori. We want to have N as small as possible, while achieving $x_{N,1} \approx x_{N,4} \approx 0.0$ and maintaining them near zero for all times greater than N.

Backpropagation through time (BPTT) is an efficient method of computing derivatives in general ordered computational structures including neural networks [9]. Derivatives are to be used for parameter optimization. To distinguish such derivatives from ordinary partial derivatives, we use the Werbos notation in which $\mathrm{F}\_^{q}z$ denotes an ordered derivative of some quantity q with respect to z. To derive the backpropagation equations, the forward propagation equations of the closed-loop system (i.e., the discrete-time disease model together with the neural network) are considered in reverse order. From each equation we derive one or more backpropagation expressions, according to the principle that if $a = z(b, c)$, then $\mathrm{F}\_^{q}b \mathrel{+}= \frac{\partial z}{\partial b}\mathrm{F}\_^{q}a$ and $\mathrm{F}\_^{q}c \mathrel{+}= \frac{\partial z}{\partial c}\mathrm{F}\_^{q}a$. The C-language notation "+=" indicates that the quantity on the right hand side is to be added to the previously computed value of the left hand side.

After running the forward propagation equations from $k = 1$ to $k = N$, we employ BPTT(N)[1] to compute derivatives $\mathrm{F}\_^{J}\mathbf{w}$ with the following algorithm:

1) Initialize $\mathrm{F}\_^{J}\mathbf{w} = 0$;

2) Initialize $k = N$ and $\mathrm{F}\_^{J}x_{N+1,j} = 0$ for all $j = 1, 2, 3, 4$. Initialize $\mathrm{F}\_^{J}x_{k,j} = 0$ and $\mathrm{F}\_^{J}u_{k,j} = 0$ for all $j = 1, 2, 3, 4$ and $k = 1, 2, ..., N$;

3) Compute for all j

$$\mathrm{F}\_^{J}x_{k,j} \mathrel{+}= \frac{\partial U_{k,j}}{x_{k,j}} \tag{6}$$

$$\mathrm{F}\_^{J}u_{k,j} \mathrel{+}= \frac{\partial U_{k,j}}{u_{k,j}} \tag{7}$$

4) Compute for all combinations of $m = 1, 2, 3, 4$ and $j = 1, 2, 3, 4$

$$\mathrm{F}\_^{J}x_{k,j} \mathrel{+}= \frac{\partial f_m(\mathbf{a}, \mathbf{x}_k, u_{k,m})}{x_{k,j}} \mathrm{F}\_^{J}x_{k+1,m} \tag{8}$$

5) Compute for all j

$$\mathrm{F}\_^{J}u_{k,j} \mathrel{+}= \frac{\partial f_j(\mathbf{a}, \mathbf{x}_k, u_{k,j})}{u_{k,j}} \mathrm{F}\_^{J}x_{k+1,j} \tag{9}$$

6) Compute for all combinations of m and j

$$\mathrm{F}\_^{J}x_{k,j} \mathrel{+}= \frac{\partial g_m(\mathbf{b}, \mathbf{x}_k, \mathbf{w})}{x_{k,j}} \mathrm{F}\_^{J}u_{k,m} \tag{10}$$

7) Compute for all j

$$\mathrm{F}\_^{J}\mathbf{w} \mathrel{+}= \frac{\partial g_j(\mathbf{b}, \mathbf{x}_k, \mathbf{w})}{\mathbf{w}} \mathrm{F}\_^{J}u_{k,j} \tag{11}$$

8) Decrement $k : k = k - 1$; continue from step 3 until reaching $k = 1$.

The algorithm 1–8 described above is the main element of our training procedure to obtain optimal $\mathbf{w}^*$. We begin the procedure by initializing weights $\mathbf{w}$ to problem-dependent values which result in stable initial controls for some number of time steps for a range of initial $\mathbf{x}(0)$ (in this problem it is sufficient to initialize $\mathbf{w}$ to small random values in the range ± 0.1). Ordered derivatives $\mathrm{F}\_^{J}\mathbf{w}$ accumulated during N loops through the steps 3–8 can be used to update weights $\mathbf{w}$. In this work we collect $\mathrm{F}\_^{J}\mathbf{w}$ from several trajectories of length N beginning with random initial $\mathbf{x}(0)$, before we update the network weights by a gradient-based method[2]. After one update we begin the algorithm anew from step 1 until approximate convergence of weights.

When the model parameters $\mathbf{a}$ and $\mathbf{b}$ in (1) are known with some uncertainty, we can apply what is called *multi-stream* method [10]. In essence, it is training the same NN simultaneously on several data streams generated by models with different values of $\mathbf{a}$ and $\mathbf{b}$. While conceptually similar to training feedforward networks on small batches

[1]This is a form of BPTT truncated after N time steps.

[2]Though various algorithms can be used, we experimented successfully with the steepest descent and the extended Kalman filter (EKF) algorithms.

of data, the multi-stream training is especially effective for networks with internal feedback. In some of our experiments discussed below, we employ such networks and use separate vectors $\mathrm{F}^{J}\mathbf{w}$ for data streams from different models of disease dynamics to update the same vector of weights $\mathbf{w}$.

IV. Example Therapies

We discuss three examples below. The initial pathogen concentration $x_1(0)$ in all examples is chosen such that the untreated (uncontrolled) case would be lethal.

A. Scalar Controls

We first consider the simplest control strategy for (1) when individual controls u_i, $i = 1, 2, 3, 4$, are applied separately. Our weighting coefficients q_i and r_j in (4) are set to the unity (r_j is one for the active control, and zero for the rest).

For all scalar control cases, applying the pathogen killer u_1 alone turns out to be the most effective way to fight the disease. Figure 2 shows the disease progress in this case. Using the algorithm of Section III with $N = 300$, we train a feedforward NN with four bipolar sigmoidal nodes in its only hidden layer and a linear positive output (its output is zero if the total nodal input is negative). We can control the process efficiency (how quickly the pathogen concentration decreases and the organ heals) by increasing q_1.

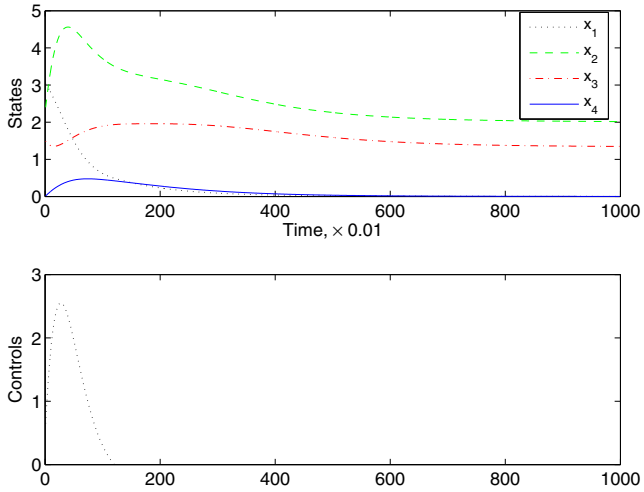

Fig. 2. Disease progress when the pathogen killer u_1 is the only active control. The initial pathogen concentration $x_1(0)$ in all Figures is chosen such that the untreated (uncontrolled) case would be lethal.

When other controls u_2, u_3 and u_4 are applied to (1) separately, they are not as quick at healing the organ as u_1. This is understandable as they are indirect controls. For example, the health enhancer u_4 helps the organ to stay healthy, so its immune system operates optimally.

B. Multiple Controls

From the standpoint of modern medical treatment, it is important to consider the effect of complex therapeutic agents, i.e., the agents which attack the disease through as many channels as possible. In Figure 3 we show our results of disease control with all control variables applied at the same time u_i, $i = 1, 2, 3, 4$. Our controller NN is recurrent. It has one fully recurrent hidden layer consisting of four nodes and four linear positive outputs. In our training algorithm we use $N = 800$.

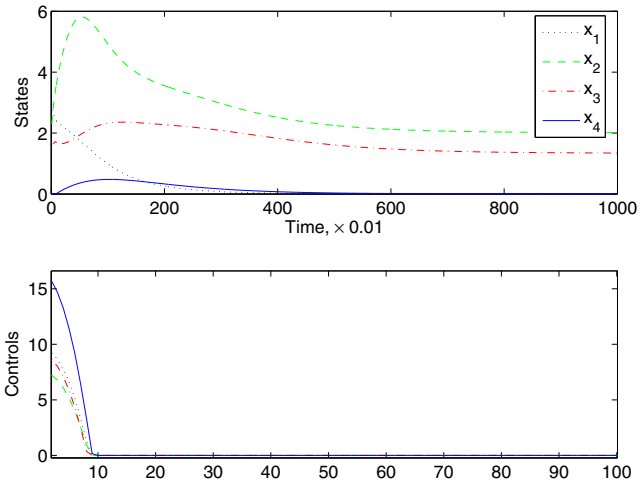

Fig. 3. Our results with all the controls active and the full state feedback. The bottom panel shows controls for the first 100 time steps. It is interesting that the largest control is the health enhancer u_4.

Rather than utilizing the full state feedback for our scalar-output neurocontroller, it is interesting to consider a more realistic partial state feedback. We can compute the observability Grammian [11] for various combinations of measurements. If all or any three components of the state are measured, the Grammian is nonsingular, and the system is fully observable. No single scalar measurement suffices. However, four out of the six pairs of measurements are sufficient for complete observability: (x_1, x_4), (x_1, x_3), (x_3, x_4), (x_1, x_2).

Figure 4 shows the results of disease control with multiple controls and the (x_1, x_4) pair as the measured inputs. The NN architecture and training algorithm used are the same as in Figure 3.

It should be noted that the training process in all our experiments is quite quick, resulting in a suitable controller within first 50 epochs of training. We observed that controllers obtained earlier in training usually have smaller maxima of controls compared with controllers obtained later in training (cf. Figures 2 – 4). (A similar effect can also be obtained if the penalty weights $q_i \gg r_j$ in (4).) Due to the integral nature of (4), their performance is not necessarily very different though.

C. Robust Control

In Section II we specified model parameter $\mathbf{a}$ and $\mathbf{b}$. All previous experiments assumed that these parameters are known precisely. Of course, this is not the case in reality. We can impose uncertainty on all parameters of the model (1) and see whether we can create a neurocontroller robust to such uncertainties.

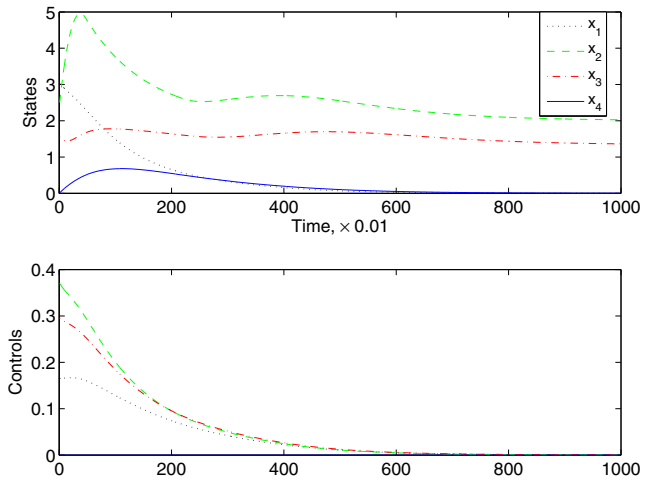

Fig. 4. Our results with all the controls active and the partial state feedback. The solution found by our training algorithm has $u_4 = 0$.

We specify $\pm 5\%$ uncertainty on the values of $\mathbf{a}$ and $\mathbf{b}$, i.e., a uniform distribution on each a_{ij} and b_j centered around their nominal values given in Section II. To train our recurrent neurocontroller, we initialize five models (1) with different values of parameters and employ the algorithm of Section III with vectors $\mathrm{F}^J\mathbf{w}$ for data streams from different models to update the same vector of NN weights $\mathbf{w}$ (multi-stream approach).

Figure 5 shows our results for robust neurocontroller. We use the same NN architecture and training algorithm as those employed to produce Figure 3. As an additional complication, we impose on all state variables a small level of uniform measurement noise ($\pm 5\%$ of signal level). We can see that the controller handles both parametric and measurement uncertainties quite well.

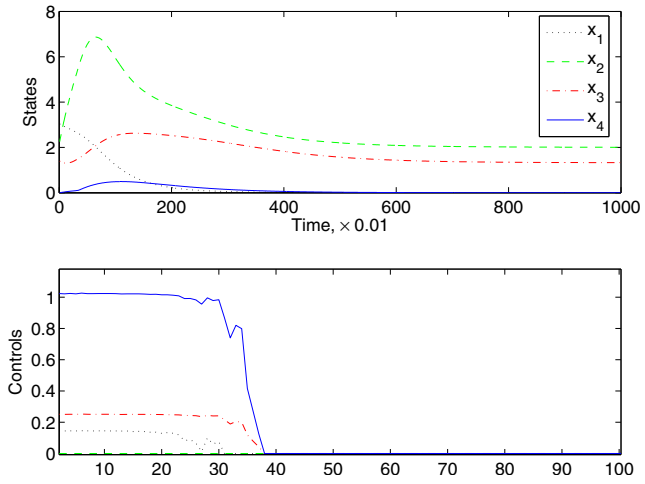

Fig. 5. Our results with robust neurocontroller. The bottom panel shows controls for the first 100 time steps. Noise influence can be seen as jagged lines of the controls. It is interesting that, given the choice of controls, the NN may prefer to ignore one of them ($u_2 = 0$); cf. Figure 4.

V. Conclusions

We study a simple nonlinear model which illustrates the characteristic behavior of various diseases and the immune system reaction. Our simulations in Section IV demonstrate that, while the patient is in the life-threatening state initially, she recovers fully when we apply our feedback neurocontrollers. Our results are also consistent with those in [6]. While current medical treatments usually take into account the feedback component, none of them assumes as frequent feedback as present in our design. As a result, our treatment times are expected to be much shorter than those of conventional treatments.

We show that various feedback control strategies are possible and effective. We also discuss how to make the "patient controller" more robust with respect to parametric and measurement uncertainties.

Future work will consider subsampled control, as well as hard control constraints to limit the dosage.

References

[1] R. E. Bellman, *Mathematical Methods in Medicine*, Singapore: World Scientific, 1983.

[2] G. W. Swan, "Optimal control applications in biomedical engineering - A survey," *Optimal Control Applications and Methods*, vol. 2, no. 4, pp. 311–334, 1981.

[3] A. Novak and G. Feichtinger, "Optimal treatment of cancer diseases," *International Journal of Systems Science*, vol. 24, no. 7, pp. 1253–1263, July 1993.

[4] J. M. Bailey, W. M. Haddad, and T. Hayakawa, "Closed-loop control in clinical pharmacology: paradigms, benefits and challenges," *Proc. American Control Conference*, Boston, MA, June 30 - July 2, 2004, pp. 2268–2277.

[5] A. Asachenkov, G. Marchuk, R. Mohler, and S. Zuev, *Disease Dynamics*, Boston: Birkhauser, 1994.

[6] R.F. Stengel, R. Ghigliazza, N. Kulkarni, and O. Laplace, "Optimal control of a viral disease," *Proc. American Control Conference*, Arlington, VA, June 25 - 27, 2001, pp. 3795–3800.

[7] D. V. Prokhorov, G. V. Puskorius, and L. A. Feldkamp, "Dynamical Neural Networks for Control," *A Field Guide to Dynamical Recurrent Networks* (Chapter 16), Edited by J. Kolen and S. Kremer, IEEE Press, 2001, pp. 257–289.

[8] D. V. Prokhorov, "Optimal Neurocontrollers for discretized distributed parameter systems," *Proc. American Control Conference*, Denver, CO, 2003, pp. 549–554.

[9] P. J. Werbos. "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[10] L. A. Feldkamp and G. V. Puskorius, "Training controllers for robustness: Multi-stream DEKF," *Proc. IEEE International Conference on Neural Networks,* Orlando, 1994, pp. 2377–2382.

[11] W. L. Brogan, *Modern Control Theory*, 3rd ed., Englewood Cliffs, NJ: Prentice-Hall, 1991.