ADAPTIVE FUZZY CONTROLLER FOR STATE-FEEDBACK OPTIMAL CONTROL

Hugues Bersini

IRIDIA - CP 194/6
Universite Libre de Bruxelles
50, av. Franklin Roosevelt
1050 Bruxelles - Belgium
email: bersini@ulb.ac.be

Abstract Whatever non-linear universal approximator, easy and fast to tune, robust enough and inherently parallel is of great interest for adaptive control as a additional way of identifying and controlling non-linear processes. Among them, fuzzy models present a singular Janus-faced: On one hand, they are knowledge-based software environments constructed from a collection of linguistic IF-THEN rules, and on the other hand, they realize nonlinear mappings which have interesting mathematical properties like "low-order and local interpolation" and "universal function approximation". The locality inherent to the way fuzzy models decompose a problem into fuzzy regions makes the design of these models a time-consuming activity especially when facing multi-variables problems. This paper first discusses how to provide fuzzy models with the capacity to automatically self-tune their parameters by using well-known gradient-based algorithms. Then the temporal extension of the gradient-based method which allows the automatic tuning of the fuzzy controller parameters over a temporal horizon will be presented and illustrated for an optimal control and a time-minimization simple problems.

Keywords: Neural control, Radial base function networks, Fuzzy control, Backpropagation algorithms, Optimal control, Gradient methods.

1. INTRODUCTION

Whatever non-linear universal approximator, easy and fast to tune, robust enough and inherently parallel is of great interest for adaptive control as a additional way of identifying and controlling non-linear processes. Besides their biological appeal and their roots to cybernetic and behaviourism, this universal modeling capacity largely explains the current and increasing attention raised by the use of neural nets in the field of control. For several years neural net non-linear approximators were limited to the so called Multi-Layer Perceptrons (MLP) characterized by units with monotonous activation function (in general sigmoïd) then covering the input space globally. More recently an alternative family of neural approximators called Radial-Basis Functions (RBF), presenting local (in general Gaussians) activation function with restricted field of activation, have appeared and their use for the adaptive control of low-dimensional processes instead of MLP have been increasingly recommended. Their basic qualities is that they are faster to tune, and due to the linear structure of the output level, easy to link with the linear control developments. Parallely fuzzy controllers such as Mamdani or Tagaki and Sugeno's (1985) controllers, which originate not from the cybernetic but rather from the knowledge and rule-based tradition of artificial intelligence, have gained a comparable popularity for the control of non-linear processes.

As a first consequence, today the debate is vivid between the adepts of fuzzy controllers and neural nets on the respective qualities of each method when applied on the same type of applications: classification, function identification and adaptive control. The confusion pervading such a debate has not been alleviated by the explosive development of a new field labeled neuro-fuzzy where features typically associated with the architecture and functions of NN like multilayer architecture or the use of backpropagation have been imported into the fuzzy community. A first aim of the paper is to help clarifying the current situation by showing that there are two visions of fuzzy models, lighting respectively the "readability" or the "performance". The first vision is clearly the most innovative one by attempting to harmonize the continuous nature of the mathematical reality with the discrete nature of the human language. "Readability" turns out to be the key property. When adopting the second vision (the one adopted in this paper), fuzzy models become, among a lot of other methods (including Normalized-RBF, CMAC, B-splines network, Locally Weighted Regression, Kernel Regression Estimator, and Neural Mixture of Neural Experts), an interesting "gray-box" way to approximate unknown non-linear mapping by a smooth combination of simple and local models (a classical strategy called "gain scheduling" in process control).

Whatever vision adopted, how to exploit the adaptive nature of fuzzy models for optimal control will be the object of the two last sections introducing the Back-Propagation-Through-Time (BPTT) algorithm. BPTT is an extension of the gradient-based static optimization of an universal approximator whenever this latter is used in a dynamical context i.e. with the error computed over a given temporal horizon. This algorithm has been derived and used by Werbos (1990), Nguyen & Widrow (1990), Parisini & Zoppoli (1991), Plumer (1993) and Suykens, De Moor & Vandewalle (1994) for the automatic tuning of neurocontroller aiming at minimizing a control cost either given over a certain temporal horizon or delayed in time. As the sections will show, the easiest and cleanest way to derive the algorithm is by introducing Lagrange multipliers and performing a gradient descent in the parameter space. This involves a backward recursion (the adjoin equations) necessary to compute the Lagrange multipliers. However the resulting algorithm, although working in a satisfactory way, can rapidly consume too much computing time and memory (this consumption increasing with the number of time steps constituting the temporal horizon). A simplified (and less consuming) version of the algorithm will be proposed in which the principle of optimality, which is at the basis of the dynamic programming method, is transposed into the BPTT backward computation.

In the last section the use of BPTT will be illustrated for optimizing fuzzy controllers on two classical and elementary process control applications: the "Rendez-Vous" and the "Mountain-Car" problems. For the second application in which the control objective is "time-minimization", we will show why and how it is necessary to extend the BPTT algorithm. Since the optimal solution of a lot of time-minimization problems is a control law showing bang-bang profile, only local type of approximator (like among others RBF and Fuzzy), well-known to behave better for approximating low-dimensional complicated and rugged functions, can provide this optimal solution.

2. ONE VISION OF FUZZY MODELS

Suppose a generic multilayer input/output mapping: **X-->Y** for handling problems of function identification of dimension N: $\mathbf{X} = (x_1, .., x_N)$ (in the following we will suppose for clarity the output space Y to be one-dimensional). The hidden layer is composed of K functions $Gl(\mathbf{X})$ (l=1..K) with "bounded support", $Gl(\mathbf{X}_c^l) = 1$ for $\mathbf{X}_c^l$ the center of the lth support, $0 \leq Gl(\mathbf{X}) \leq 1$ and $Gl(\mathbf{X})$ decreases when getting away from the center. Moreover the further choice is made of Gl(X) being factorizable on the axis: $Gl(\mathbf{X}) = \prod_{i=1}^{N} Gl(xi)$.

The next layer is composed of a set of K N-dimensional functions $Fl(\mathbf{X})$ (called the "local actor") thus the same number of F as the number of G and each F is respectively associated to one G which serves to mark its zone of influence. The last layer computes the final output Y by a certain global combination of the G-F functions. Fuzzy inference systems are obtained when the Gl(xi) (called "membership functions" in the fuzzy community) are taken to be triangles, trapezoids or Gaussians function centered in $\mathbf{X}_c^l$ (here for the comparison we will restrict to Gaussians: $Gl(xi) = \exp -(\frac{xi - x_{ci}^l}{d})^2)$ and the final combination is the weighted average, in which the influence of each "local actor" Fl is weighted by the value of its associated Gaussian:

$$Y = \frac{\sum_{l=1}^{K} Gl(\mathbf{X}) Fl(\mathbf{X})}{\sum_{l=1}^{K} Gl(\mathbf{X})} \qquad (1)$$

In terms of fuzzy logic each couple Gl-Fl is the expression of a lth rule such as:

$$\text{IF } x_1 \text{ is } A_1^l \text{ and } x_2 \text{ is } A_2^l \text{ and } \ldots\ x_n \text{ is } A_n^l \text{ THEN } Y = F(X)$$

where the A_i^l are the linguistic labels of the variable i (small, average, big...) associated to rule l.

Fuzzy controllers have been recognized as an attractive alternative to classical control schemes for the low-cost and facilitated design of the control laws of imperfectly known, nonlinear and complex processes. Their intrinsic linguistic nature makes easier the exploitation and integration of qualitative knowledge of the control laws, constraints and objectives. As a matter of fact, it is this possible writing of fuzzy models in terms of linguistic production rules that requires the functions G to be factorizable provided the logical connective AND be numerically transposed into a product. Relaxing this readability would allow to enlarge the set of acceptable G (including ellipsoid no longer parallel to the axis).

Classically either F is a constant α^l or F is a linear sum:

$$Fl(\mathbf{X}) = \sum_{i=1}^{N} \alpha_i^l x_i + \alpha_{n+1}^l$$

It is easy to see (with others (Jang & Sun, 1993), (Nie & Linkens, 1993) in the fuzzy community) that by taking the combination to be the weighted sum and no longer the weighted average and by restricting each function F to be a constant, the outcome is a perfect equivalence between RBF and fuzzy controllers. Now these two small structural additions which characterize fuzzy models can have surprising implications on the respective interpretation of how the two methods work. In RBF (with constant F and weighted sum combination) the first layer (like for classical MLP) treats the input data so as to project them into a new space in which the final operation, whatever task it is: identification or classification, can be accomplished in the simplest linear way (this is the basic message of Cover's theorem underlying RBF classifiers). The advantages are manifold: use of fast clustering algorithms for locating the Gaussians, acceleration of any form of gradient-based algorithm due to the locality of the Gaussians (learning in some place does not unlearn in some others, this faster learning is a key property that favours the use of RBF instead of MLP for on line adaptive control), use of one shot algorithm for linear optimization of the second layer, proofs of convergence and proofs of stability for adaptive control (when the adaptivity is restricted to the output linear layer), reduced sensitivity to the order of presentation, etc. Although the first layer acts in the problem input space, the second linear layer now acts in an abstract space without clear correspondence with the problem space.

With fuzzy models the story is different. The Gaussians mark the zone of the input space where the local actors F (the output of the fuzzy rules) will perform their simple function (we suppose in the following to keep the F linear although they can have whatever simple shape). So the Gaussians naturally entail a problem decomposition and each F will face the problem in a restricted zone of the input space: a local approximator. In contrast with RBF the second linear layer always acts in the problem space but in a restricted part of it: F can be seen as a local perceptron or a local linear approximator, whose zone of influence is marked by its associated Gaussian and a smooth transition is guaranteed between the effects of different F acting in distinct but overlapping zones. The difference between RBF and fuzzy models in function identification is very sharp: fuzzy models locally approximate a function in a linear way and a smooth interpolation between two linear approximators is based on the respective overlap of a precise input with the intersecting "fuzzy" partitions. In RBF what make the local approximation are clearly the Gaussians and not the constant weights.

3. GRADIENT-BASED ALGORITHMS FOR OPTIMAL CONTROL

Now whatever universal approximator (supplied with a gradient-based automatic tuning of its parameters: synaptic weights or membership functions) to be used for capturing the non-linear control law, the gradient-based algorithm can be extended for optimal control. For simplicity, we will restrict the following analysis to SISO processes.

Suppose the process to be described by its state-space equation:

$$x(n) = G[x(n-1), u(n-1)] \qquad (2)$$

where x(n) is the state of the process at time n and u(n) the control actions. The following control cost needs to be minimized:

$$J = \frac{1}{2} \sum_{n=t0}^{t0+H-1} [x(n+1)^T F_{n+1} x(n+1) + u(n)^T H_n u(n)]$$

where Fn and Hn are symmetric definite positive matrices. The purpose here is to stabilize the process around x=0 with minimal cost on the control actions. We assume a state-feedback class of controller with the control law given by u(n)=U[x(n),x0] where U is a differentiable vector function. Similar problems have been addressed by Werbos (1990), Nguyen & Widrow (1990), Parisini & Zoppoli (1991), Plumer (1993), Suykens, De Moor & Vandewalle (1994) and Saerens, Renders & Bersini (1995). The purpose of the learning algorithm is to train fuzzy controllers to provide the optimal control law for all initial conditions in a given operating region u(n)=N[x(n),x0;w] where N is the mapping computed by the universal approximator and w its parametric matrix. Without loss of generality, the time origin is redefined as t0=0. In order to find the optimal control sequence (u(0), ...,u(H-1)), the Back-Propagation-Through-Time (BPTT) algorithm can be derived by introducing the corresponding Lagrange function:

$$L = \frac{1}{2} \sum_{n=0}^{H-1} [x(n+1)^T F_{n+1} x(n+1) + (u(n)^T H_n u(n)]$$
$$+ \sum_{n=0}^{H-1} \lambda_n^T (x(n+1) - G[x(n), u(n)] +$$
$$\sum_{n=0}^{H-1} \mu_n^T (u(n) - N[x(n), x_0; w]) \qquad (3)$$

where λn and μn are vectors of Lagrange multipliers.

It is very easy to show that by considering all variables as independent and deriving the necessary conditions of optimality with respect to u(k), x(H), x(0),x(k), λk and μk, the BPTT algorithm naturally follows. The general procedure to adjust the parameters by gradient descent is

(1) The process is running from k=0 to k=H and the values of the x(k) and the u(k) are memorized for any time.
(2) The conditions of optimality allow to compute recursively (the adjoin equations) the Lagrange parameters λk (k=0,..,H-1) and μk (k=0,..,H-1):

$$\lambda_H = -F_H x(H)$$

$$\mu_k = -H_k u(k) + \left(\frac{\partial G_{k+1}}{\partial u(k)}\right)^T \lambda_k$$

$$\lambda_{k-1} = -F_k x(k) + \left(\frac{\partial G_{k+1}}{\partial x(k)}\right)^T \lambda_k + \left(\frac{\partial N_k}{\partial x(k)}\right)^T \mu_k$$

(3) The parameters are updated by gradient descent: $\Delta w = -\eta \frac{\partial L}{\partial w} = \eta \sum_{n=0}^{H-1} \left(\frac{\partial N_k}{\partial w}\right)^T \mu_k$ where the term $\partial Nk/\partial w$ is computed by using backpropagation algorithm for the neural nets or its counterpart for fuzzy systems.

Some prior knowledge of the process is needed for computing $\partial G/\partial u$ and $\partial G/\partial x$. This prior knowledge can be obtained directly from an analytical model of the process or from a preliminary approximated modeling of the process.

An application of the BPTT method to nonlinear plant control is reported in Beaufays, Abdel-Magid & Widrow (1994). Moreover it has been shown by Saerens, Renders & Bersini (1995) that the backward recursive computation that allows the computation of the gradient can be transformed into a forward algorithm that can be carried out on-line, assuming that the adaptation rate of the weights is slow. This backward recursive form of the neural net backprop learning can easily be re-discovered by (initially proposed by Nguyen and Widrow) separating the successive time steps, and assuming that the control for each time step is the responsibility of a distinct approximator (so at the end as many approximators as stages for the controller, n for instance). Now in order to fall on the same learning equations for one and only one approximator, it suffices to add the n errors each approximator is responsible for first to adjust the parameters. In general the backward recursion (the adjoin equations) necessary to compute the Lagrange multipliers can rapidly consume too much time and memory, this consumption increasing with the number of time steps constituting the temporal horizon.

There are two reasons that allow us to simplify further the BPTT algorithm. First when the successive time steps are explicitly given, the optimal control law is just function of the initial conditions and the current time step, thus no longer dependent on the current state. Besides the dynamic programming principle of optimality tells you that, provided a cumulative cost needs to be optimized, once the final part of a job is optimized, the precedent part can be optimized in considering the work already done at its best for this final part. Therefore the successive optimization can be decomposed and made independent i.e. it is no longer necessary to try to optimize the output of the nth approximator with respect to the parameters of all previous approximators: a complete temporal separability can be preserved with benefic effect to reduce the learning algorithm to:

$$\mu_k = -H_k u(k) + \left(\frac{\partial G_{k+1}}{\partial u(k)}\right)^T \lambda_k \quad (4)$$

$$\lambda_{k-1} = -F_k x(k) + \left(\frac{\partial G_{k+1}}{\partial x(k)}\right)^T \lambda_k \quad (5)$$

The parameters are again updated by gradient descent:

$$\Delta w = -\eta \frac{\partial L}{\partial w} = \eta \sum_{n=0}^{H-1} \left(\frac{\partial N_k}{\partial w}\right)^T \mu_k. \quad (6)$$

Notice that in comparison with the previous original algorithm it is indeed what makes the link between the different time step which has been canceled: the derivative of the output of approximator k with respect to the output of approximator k-1 i.e $\partial Nk/\partial x(k)$. Moreover the parameter μk although left here for sake of clarity, is actually no longer necessary since its recursive role has been eliminated. This leads to an important algorithmic simplification.

4. THE "RENDEZ-VOUS" AND THE "MOUNTAIN-CAR" PROBLEMS

These two simple and well known optimal control problems will be now presented in order mainly to illustrate experimentally how BPTT optimizes adaptive fuzzy controllers.

The "Rendez-Vous" problem is one of the simplest optimal control applications characterized by a delayed feedback. It is a one dimensional problem in which an immaterial point (with coordinate y) leaving a departure point y_0 with an initial velocity v_0 must reach the axis origin (the place for "Rendez-Vous") with a zero velocity, in a finite amount of time step kf and minimizing the consummation given by the integral of the successive accelerations.

The problem is defined as follows: k: current iteration; y(k): the point position; v(k): the point velocity; a(k): point acceleration (and the actual control); x(k): state vector={y(k), v(k)}; x(0): initial state vector and kf: final iteration. The cost to be minimized is given by: $J = \frac{1}{2}\vec{x}(k_f)^T \vec{x}(k_f) + \frac{1}{2} c \sum_{0}^{kf-1} a(k)^2$ with c weighting the consumption saving requirement. The process equations in their discrete form are:

$$y(k) = y(k-1) + v(k-1)\Delta t + a(k-1)\Delta t^2/2$$
$$v(k) = v(k-1)+a(k-1)\Delta t$$

The acceleration is the output of a local approximator which receives as inputs the position and velocity. Here the control approximation is achieved by a fuzzy controller in which all parameters (the Gaussians centers and bases as well as the linear weights) are automatically adjusted by gradient descent. All the Jacobians are derived from the analytical equations. Besides all the Jacobians can be approximated by their sign following a further simplification strategy already discussed in (Saerens, Renders and Bersini, 1995). In their parking of the truck backer-upper, Nguyen and Widrow (1990) used a second approximator for emulating the process and deriving the Jacobians. Fig.2 shows the error in time and the final trajectory for c=1000, kf=100 and three different initial velocities.

Fig.1. The acceleration profile for the Rendez-Vous Problem for c=1000 and kf=100.

The mountain car problem, is a slightly different application, still characterized by two variables y (the height) and v (the velocity) but now described by the following state equations:

$$y_{k+1}=y_k+v_k\,\Delta t+(u_k+g\sin y_k)\Delta t^2/2$$

$$v_{k+1}=v_k+(u_k+g\sin y_k)\Delta t$$

In addition, the acceleration must remain in the interval [-umax,umax]. It is a “time-minimization” problem in which the car must reach the top of the mountain in a minimum of time steps. Since the car can't receive enough power to get until the top, it must initially get away from the top and make backward climbing for absorbing enough potential energy to compensate for the lack of control power.

In order to minimize the duration of the control sequence, Plumer (1993) has proposed to extend the standard BPTT (the original version) to account for this new temporal constraint. The basic idea is that minimizing a cost now given by:

$$J = \frac{1}{2}\sum_{n=t0}^{t0+H-1}\left[x(n+1)^T F_{n+1}x(n+1) + u(n)^T H_n u(n)\right] + \alpha(H-1)$$

with respect also of the parameter H can be decomposed in two successive and separated minimizations i.e. min J(u(t0),..,u(H-1);H) = min J(u(t0),..,h(Hmin-1);Hmin). The resulting new algorithm differs from BPTT in that, during the forward controlling phase, the cost J is computed at each time step until J comes to a minimum. Once this minimum is attained, the backward classical recursive past is released just like in BPTT. Then in contrast to the standard BPTT, the forward controlling run is stopped and J is evaluated (including the time-minimization term $\alpha(H-1)$) at the time-step which minimizes this cost. Plumer shows very good experimental results achieved by this method when controlling a boat navigating in the river against a linear current. Now one particularity of this control application is that the boat must point in the right direction from the very beginning of its control sequence. An interesting counterexample of such a control situation is indeed the mountain-car problem where the optimal solution forces the agent to initially move in the opposite direction. Although further experimental work would have been necessary, experimental difficulties were met when applying Plumer's approach in the mountain-car experiment. These difficulties were however simply resolved by applying one possible alternative extension of BPTT very similar in spirit to Plumer's proposal but easier to implement and to make successful.

Following several attempts and as a consequence of the bang-bang profile of the optimal solution, the only possible neurocontrollers we could use for this application had to be a local type, here a RBF but a fuzzy controller would has been as well. We added a sigmoid activation function at the output layer to saturate the acceleration both positively and negatively. The acceleration was then given by $u_k = \dfrac{1 - \exp[\sum_{i=1}^{N}\rho_i w_i]}{1 + \exp[\sum_{i=1}^{N}\rho_i w_i]}$ with N the number of radial functions, ρ_i the output of the ith radial function and w_i the weight connecting this output to the acceleration.

It is very easy to show that in the backward pass the output weights (the only parameters to be automatically adjusted) are modified as follows:

$$\Delta w_j = 2\rho_j \sum_{k=0}^{k_f} \frac{\exp[\sum_{i=1}^{N}\rho_i w_i]}{(1 + \exp[\sum_{i=1}^{N}\rho_i w_i])^2}\mu(k) \qquad (7)$$

with kf the number of time steps.

The total cost to be minimized is given by: $L = \beta k_f^2+y(k_f)^2$ with $y(k_f)$ the final position of the car.

Fig.2. The Mountain-Car Problem: The successive solutions resulting from a progressive decrement of the number of time steps

The control strategy proposed as an alternative to Plumer's one was very simple: running successive BPTT by gradually decrementing k_f i.e. the number of time steps and this until no solution can be found any longer. It is illustrated in fig.2 where the position is shown as a function of time for successive decrement of k_f. The only prior requirement is to guess a maximum initial value for k_f. Therefore a progressive decrement of the number of time steps until no solution is possible any more can be an interesting alternative to Plumer's approach when the control application to optimize in time raises some further difficulties.

5. CONCLUSIONS

A first message delivered in this paper is the today existence of two visions of fuzzy models, first as a linguistic translation of a numerical mapping and then as a powerful multi-local-model gray-box approximator. These two visions can be incompatible to some extent i.e. the search for a readable model can be done at the expense of the computational cost, and above all at the expense of the performance in approximating real data. The growing interest in neuro-fuzzy models assumes an important responsibility not only for deepening the gap between these two visions but also for "fuzzyfying" in which context and for what reason fuzzy models are used.

If readability is the main objective, first a precise understanding of what is meant by "readability" is necessary, then a whole new set of optimization methods, very dissimilar from their implementation in neural models, must still be derived to complement the human expert knowledge in the automatic generation of fuzzy models. Also more discussion would be welcome on the necessary compromise to reach between a satisfactory approximation and a readable model. If optimal performance in control, function identification or classification is the main objective, then an exhaustive and deep comparison with closely related methods (local-based and multi-models) has to be done, for realistic problems, in terms of pure performance, computational cost and practicability.

In the last part, the gradient-based optimization method, used to tune the approximators parameters (whatever synaptic weights or membership functions), has been extended for optimal and time minimization controls. BPTT and Time-Optimal-BPTT have been described and applied on two very simple problems. For the last application and due to the bang-bang profile of the control law, only RBF and fuzzy controllers were likely to provide a solution. Besides their intrinsic properties stated above, this appears as a additional reason to favour the use of "gray-box local approximator" for adaptive control showing rugged profile.

REFERENCES

Beaufays F., Abdel-Magid Y. & Widrow B., 1994, Application of neural networks to load-frequency control in power systems. *Neural Networks*, 7 (1), pp. 183-194.

Jang R. J.S. & Sun C.-T., 1993, "Functional equivalence between radial basis function networks and fuzzy inference systems". *IEEE Transactions on Neural Networks*, 4 (1), pp. 156-159.

Nguyen D. & Widrow B., 1990, Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, 10 (3), pp. 18-23.

Nie, J. and D.A. Linkens (1993). Learning Control Using Fuzzified Self-Organizing Radial Basis Function Network - IEEE Transactions on Fuzzy Systems, Vol.1, No4.

Tagaki, T. and M. Sugeno 1985: Fuzzy identification of systems and its application to modelling and control. *IEEE Trans. Syst. Man Cybern.*, vol SMC-15, no. 1, pp 116-132.

Parisini T. & Zoppoli R., 1991, Backpropagation for n-stage optimal control problems. *Proceedings of the IEEE International Joint Conference on Neural Networks*, Singapore, pp. 1518-1529.

Plumer E., 1993, Time-optimal terminal control using neural networks. *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, pp. 1926-1931.

Saerens M., Renders J-M. & H. Bersini, 1995, Neural Controllers based on Backpropagation Algorithm - in *IEEE Press Book on Intelligent Control*, M. Gupta & N. Sinha (Eds) IEEE Press. pp.292-326.

Suykens J., De Moor B. & Vandewalle J., 1994, Static and dynamic stabilizing neural controllers, applicable to transitions between equilibrium points. In *Neural Networks*, 7 (5) - pp. 819-831.

Werbos P., 1990, Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78 (10), pp. 1550-1560.