Fifth IFSA World Congress (1993), 1350-1353

Optimal Tuning of a Fuzzy Controller Using Box's "Complex" Algorithm

Thomas Whalen and Brian Schott
Decision Science Department, Georgia State University
Atlanta, Georgia 30303-3083

Abstract:
A fuzzy control system typically requires "tuning," or adjustment of the parameters defining its linguistic variables. Automating this process amounts to applying a second "metacontrol" layer to drive the controller and plant to desired performance levels. Current methods of automated tuning rely on a single crisp numeric functional to evaluate control system performance. A generalization of Box's complex algorithm allows more realistic tuning based on lexicographic aggregation of multiple ordinal scales of performance, such as effectiveness and efficiency. The method is presented and illustrated using a simple inverted pendulum control system.

1. Control and Metacontrol

Figure 1 presents the basic idea of a control system. The controller compares the output of some physical process (the "plant") against an ideal value (the "control objective" or "set point") and applies a control signal to the plant. The goal is to drive the plant's output toward the control objective. In general the plant's behavior is also subject to an uncontrolled disturbance. The disturbance may be an initial difference between the plant output and the control objective, it may consist of later fluctuations that tend to push the plant's output away from the control objective, or both. (More detailed treatments of control dynamics make a distinction between the internal state of a plant and its external output. The present discussion suppresses this distinction without undue loss of generality since observable states can be mapped into the output vector while unobservable states can be approximated as disturbances.)

Figure 2 adds another element to the picture. The "metacontroller" observes the inputs and outputs of the controller. These inputs and outputs are accumulated and compared against metacontrol objectives; on the basis of this comparison the metacontroller adjusts the parameters of the controller to bring the behavior of the system as a whole closer to the metacontrol objective.

Metacontrol is almost always an important part of the design and implementation of a control system. The process begins with choosing the basic design of the controller; examples include traditional Proportional Integral Differential (PID) controllers, neural nets, and fuzzy logic control systems. In any case, the controller design will include several parameters whose values collectively specify a particular member of a general family of controllers. These parameters define a mathematical space that must be searched to find a satisfactory or optimal controller for the system in question.

The "metacontroller" in the early stages of implementation is the members of the design team themselves, aided by general purpose hardware and software. The design team specifies various prototype versions of the controller, each of which defines a point in parameter space. They then test each prototype against a real or simulated plant. The evaluation criteria for judging prototype controllers can be conceptualized in terms of two broad categories. The first category, effectiveness, concerns how well the controlled system approximated the control objective. The second category, efficiency, measures how well the controller itself performed in terms of resources consumed and unwanted side effects produced.

The design team uses information about the effectiveness and efficiency of the controller prototypes investigated so far to pick new points in parameter space to be investigated. The process continues iteratively until the system is good enough to be released.

The process described above relies very heavily on human judgment and expertise; we may call it the "manual metacontrol" paradigm. There are several good reasons to try to automate parts of the metacontrol process. One strong impetus comes from the area of adaptive control, in which the metacontroller is integrated with the controller. An adaptive system continually evaluates the effectiveness and efficiency of the control process and continually or intermittently modifies the values of the control parameters to improve them. It has become customary to refer to this updating of control parameters as "learning" by analogy with the way a human operator improves his or her control of a system with increasing experience.

Another reason to automate the metacontrol process is to be able to deliver more control systems to the market in less time. Advances in hardware and software have opened up many opportunities to market "smart" devices of all sorts, as soon as control systems exist to implement them. In this context, automated metacontrol systems become an important component of computer aided design, greatly increasing the productivity of control engineers.

A final reason for automated metacontrol is documentation. A controller that arises from a standardized search algorithm with a well defined stopping criterion may be easier to recognize as "good enough" than one which is simply the best one seen so far in an undocumented process of trial and error search.

2. Fuzzy Logic Control

Fuzzy controllers have received considerable attention, both practical and theoretical, because domain experts with no special training in control engineering can qualitatively frame fuzzy rules for narrowly defined systems. [Sugeno & Yasukawa, 1991] The general structure of such rules can often be acquired rather directly because of their linguistic flavor; nevertheless, tuning or calibrating the fuzzy variables can still be very challenging.

We consider standard fuzzy controllers which encode their knowledge as rules comprised of combinations of subrules. A typical subrule i has the form "If the value of xi is $\mathcal{X}_i$ and the value of yi is $\mathcal{Y}_i$, then the value of zi should be $\mathcal{Z}_i$." Lower case letters x and y signify the names of antecedent variables such as position and velocity; $\mathcal{X}_i$ and $\mathcal{Y}_i$ are fuzzy linguistic values describing these variables. Similarly, z and $\mathcal{Z}_i$ are a consequent variable and its fuzzy value.

The rule contains subrules $i=1,\ldots,\ell$ which are fused into the overall rule by the fuzzy operator minimum or maximum, depending on the multivalued logic employed in the system. (Minimum for material implication type logics, maximum for

Mamdani-type logics.) The term set for the fuzzy values X, Y, and Z commonly includes: Large negative; Negative; Small negative; Zero; Small positive; Positive; and Large positive. A typical subrule is "If the error angle is Small negative and the angular velocity is Small negative, then the force of the push should be Small positive." The current study uses a system that contains one rule with eleven subrules.

In operation, the fuzzy controller observes the actual data values for the antecedent variables x and y, denoted X and Y. In a practical fuzzy control system, the actual values are observed in the form of crisp numeric singletons. Also the operational controller defuzzifies the rule's detached consequent value Z into a crisp numeric singleton that is the actual control signal to the plant.

Common performance variables for mobile systems are safety, fuel economy, smoothness of ride, and speed of recovery. Performance factors of the controller itself include speed, robustness, memory needs, physical dimensions, and cost. We are concerned in this study with the effect of tuning decisions upon two performance factors: the length of time the pole remains balanced (effectiveness), and the smoothness of the control (efficiency). We attempt to optimize system performance in relation to these criteria, seeking effectiveness first and then efficiency, The methodology employed does not assume that the controllable factors and the performance variable have continuous numeric values.

3. Metacontrol for Fuzzy Logic Controllers

Researchers have proposed and tried many approaches to searching parameter spaces of controllers in general, and fuzzy logic controllers in particular. Nearly all the automated approaches involve defining a single numeric objective functional[1] or "figure of merit" to express both the effectiveness and the efficiency of the control process. Given such an objective functional, various workers have optimized it analytically [Kirk, 1970], using Response Surface Methodology [Schott & Whalen, 1992], using neural networks [Hayashi et al, 1992; Kosko, 1992; Berenji, 1992; Keller & Tahani, 1992], as well as other approaches. However, the use of a single numeric objective functional seems to work directly contrary to the principal advantages of fuzzy control systems over their non-fuzzy counterparts.

In a fuzzy control system, the control objectives themselves may be ordinal; they need not be restricted to an interval or ratio scale. Fuzzy control objectives can reflect and exploit the fact that many real situations are more or less tolerant of imprecision. For example, a fuzzy controller may strive to maximize a global assessment of "comfort" in a transportation system while a nonfuzzy system can only optimize some mathematical function combining acceleration, vibration, and noise. As a result, it seems questionable to tune a fuzzy controller using optimization procedures that attempt to estimate first and second derivatives of the degree to which the system meets its fuzzy control objectives.

Ordinal scales also facilitate lexicographic and other nonlinear approaches for dealing with multiple objectives. The use of a single numeric objective functional to capture all aspects of effectiveness and efficiency trade-offs can be problematic even in nonfuzzy control environments. And it is doubly questionable to represent the trade-offs between fuzzy effectiveness and fuzzy efficiency with a single crisp functional.

A classic algorithm, coincidently published in the same year as Zadeh's original article on fuzzy sets, provides a solution to both these problems. The algorithm is due to M.J. Box [1965; Himmelblau 1972 p.177-178]; it is called the "complex" algorithm not because it is especially intricate but because it involves a set of points in parameter space consisting of more than the minimum number of points necessary to span the space. Box called such a set of points a "complex" to distinguish it from a simplex which contains only the minimum number of points, which is one more than the number of dimensions.

The great advantage of Box's complex algorithm for tuning a fuzzy controller is that it uses only ordinal evaluations of the quality of points in parameter space. In fact, the complex algorithm can seek an optimum even when the quality of the points is incompletely ordered. (The algorithm also has some faults; when the objective is single and differentiable, derivative based approaches require fewer evaluations of the objective function [Himmelblau, 1972]. Also, the algorithm can fail when the minimum lies along a sharply curving valley.) In the present research, we extend the Box algorithm to handle mulitple ordinal objectives lexicographically.

4. Tuning Using Box's Complex Algorithm

The following discussion presents a generalization of Box's original algorithm as applied to tuning a fuzzy control system with some free parameters. The approach generalizes that of Box by explicitly considering the possibility that the quality of points might not be completely ordered. It is possible that two control systems may perform equally well within the limits of our ability to judge them. It is also possible that two control systems may be clearly different in their performance, but we are still unwilling or unable to say one is better and the other worse. This can happen when one is clearly more effective, but the other satisfies minimal effectiveness requirements and is much more efficient.

Step 1:

To begin tuning a fuzzy controller using the complex algorithm, select an initial complex of points in the mathematical space defined by the parameters of the control system to be implemented. (Box suggests that the number of points be three times the number of parameters.) Each point defines a control system; run each control system with a real or simulated plant for a standard trial period. It is important that the points span the space of parameters; this can be accomplished either by using a design matrix as in [Schott & Whalen, 1992] or by random perturbation from an initial value as suggested by Box.

Step 2:

Rank the points from best to worst with respect to the performance of the corresponding controllers. (Ties and incomparables are allowed.) For example, an engineer might rank the performance of control systems for a vehicle in terms of safety while a human factors expert ranked them in terms of comfort and an accountant ranked them in terms of cost. The final ranking might depend on safety, with ties on safety broken by trading off comfort and cost.

Step 3:

Select the worst point in the current set. If there is no unique worst point, idnetify the set of points that are not ranked better than any other point, and randomly select a point from among those. Construct a line in parameter space from the worst point to the centroid of the other points. Multiply the distance from the worst point to the centroid by an "overexpansion factor" (Box suggests 1.3), and extend the line beyond the centroid by a distance equal to the result. This defines the new candidate point in parameter space.

Step 4:

Run the corresponding control system with the real or simulated plant for a standard trial period, and compare its performance with the worst point in the complex. If the new point is better than the old worst point, replace the latter with the new point and return to step 2. If the new point is worse than the old worst point, if the two points are tied, or if the two points are not comparable, then create a new candidate point half way between the old candidate point and the centroid. Make this the new candidate point and repeat Step 4.

---

1. The term "functional," as opposed to "function," indicates that the argument of the objective functional is itself a function of time, which tracks system performance throughout the test period.

Continue these steps until the points in the complex are all within a predetermined radius of one another or until some other criterion is met. If the algorithm seems to be stuck in Step 4, check the performance of the control system defined by the centroid of the complex. If the centroid control system performs worse than any of the points in the complex, Box's algorithm cannot proceed. To get further improvement in this case, re-start the algorithm with a new complex in the vicinity of the best points seen so far.

5. Example: Tuning an Inverted Pendulum Controller

Control of an inverted pendulum has become a common benchmark problem among fuzzy researchers. A cart on a straight track is pushed with varying degrees of force according to the controller's instructions. A sensor detects the angle Θ in radians that the pole makes with the vertical. The angular velocity of the pole angle, Θ', is computed approximately based on the change in Θ. Another sensor measures the cart's position δ relative to its starting position. A pushing force Γ is applied to the cart. Θ, Θ', δ, and Γ can take positive or negative values depending on leftward or rightward orientation.

The fuzzy controller uses eleven sub-rules containing Θ and Θ' as antecedent variables, and with Γ as the consequent variable. Five terms were defined for each variable: Negatve; *Small negative*; *Zero*; *Small positive*; and Positive. All fuzzy (linguistic) variables were represented as symmetrical trapezoids. The scales of all the trapezoids on each universe of discourse were uniform relative to one another, but the scales on different universes were independent.

The controller was tuned (metacontrolled) by calibrating the scale[2] of the axes of the three universes: Θ, Θ', and Γ. Two criteria were used for optimization. The most important goal of the system, its "effectiveness," was simply to balance the pole. The system was run for a simulated period of 5 seconds, divided into 250 "ticks" of the simulation clock. If the pole angle Θ passed out of the controllable range during this time, the number of ticks remaining in the test period was reported. If the pole was still standing at the end of the simulation, this figure was equal to zero. Effectiveness scores are presented in Tables 1 and 2 in the column headed "time left;" the smaller this number, the more effective the control.

A second goal, "efficiency," was to achieve a smooth, steady balancing of the pole rather than a jittery or runaway one. (Cart-pole systems are subject to a "runaway" condition, in which the pole remains balanced at an angle while the cart accelerates continuously until it runs off the end of the track.) The second goal was represented by the product of two quantities: the integral of the absolute angle $|\Theta|$ and the integral of the absolute cart position $|\delta|$. This number is very large under runaway conditions and moderately large for an inefficient, jittery control system. Effectiveness scores are presented in Tables 1 and 2 in the column headed "instability;" the smaller this number, the more efficient the control.

The two goals were combined lexicographically. In other words, any control system that balanced the pole for a longer period ranked better than any control system that balanced the pole for a shorter period, regardless of their relative efficiency ratings. If the control systems both balanced the pole for the entire experimental period, or if they both balanced the pole for equal periods of time before losing control, then the more efficient control system ranked higher. (The actual scores used were time remaining when control was lost and a measure of inefficiency, so optimization was by minimization.)

Following Box's suggestion, we used nine original sample points to tune the three parameters of the system. The triads (Θ, Θ', and Γ scales) for each of the 9 original sample points in Table 1 were set judgmentally to include a broad range of reasonable designs. Each point specifies the scale values of the 3 variables: pole angle scale in radians, pole angular velocity scale in radians per second, and pushing force scale in newtons.

The initial nine points consisted of the controller for which all three scales equalled 2.0 plus the eight controllers formed by all combinations of:
angle (Θ) scale = .03 or 1.0 radians;
angular velocity (Θ') scale = .02 or 2.0 radians/second;
and push (Γ) scale = 1.0 or 10 newtons.

The smaller the scale for Θ and Θ', the more sensitive the controller is with respect to changes in angular position and velocity. The larger the scale for Γ, the stronger the output of the controller. At each stage of tuning, the 9 points are presented in sorted order, so the ninth row is always the worst of the 9 points currently under consideration.

Every experiment was run with a starting angle $\Theta = 0.05$, angular velocity $\Theta' = 0$, and position $\delta = 0$. Time was incremented every 0.02 seconds, cart mass was 1.0 Kg, pole mass was 0.1 Kg, pole length was 0.5 m, and acceleration due to gravity was 9.8 m/s^2. The simulation was based on differential equations provided by Hamid Berenji [Berenji, 1992]. The simulation assumed a frictionless plant and certain other simplifications, and is not intended to precisely represent a real inverted pendulum. Box's complex algorithm was implemented as a Lotus 123 spreadsheet.

Table 1 shows the original set of nine points in parameter space along with the two criterion variables for each one. At the bottom of the table the coordinates of the centroid point and the vector from the worst point to the centroid appear. In the box at the top of the table appear the coordinates of the next candidate point. The box also contains the locations where the user will enter the values of the criterion variables from the simulation using the controller defined by the candidate point.

Table 2 shows the spreadsheet after approximately 65 iterations. Note that all nine points are much closer together than in the original spreadsheet. The first of the nine points would define the system to be implemented and marketed if this were an actual design project.

References

Berenji, H. R. "A reinforcement learning-based architecture for fuzzy logic control," Int. J Approx Reasoning 6:2, pp. 267-292, 1992.

Box, M.J. "A New Method of Constrained Optimization and a Comparison with Other Methods," Computer J. 8, p.42-52,1965.

Hayashi, I.; Nomura, H.; Yamasaki, H. & Wakami, N. "Construction of fuzzy inferences rules by neural network driven fuzzy reasoning and neural network driven fuzzy reasoning with learning functions," Int. J Approx Reasoning, 6:2, pp. 241-266, 1992.

Himmelblau, D. Applied Nonlinear Programming, McGraw-Hill, New York, 1972.

Keller, J. M. & Tahani, H. "Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks," Int J Approx Reasoning 6:2, pp. 221-240, 1992.

Kirk, D. Optimal Control Theory: An Introduction, Prentice-Hall, Englewood Cliffs, 1970.

Kosko, B. Neural Networks and Fuzzy Systems, Prentice-Hall, Englewood Cliffs, 1992.

Schott, B. & Whalen, T. "Tuning a Fuzzy Controller Using Quadratic Response Surfaces," Proceedings, N Amer. Fuzzy Information Processing Society Conference, 1992.

Sugeno, M. & Yasukawa, T. "Linguistic modeling based on numerical data," Proceedings of the 4th International Fuzzy Systems Association Congress, 1991, pp. 264-267.

A preliminary version of this paper appeared as a poster session in the 1993 IEEE Conference on Fuzzy Systems.

2. *Each continuous variable's axis was discretized at 17 equidistant values. The "scale" value is the distance between adjacent points, with the median (eighth) point always anchored at zero.*

Figure 1: Basic Control System

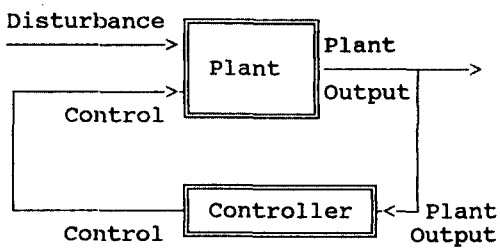

Figure 2: MetaControl System

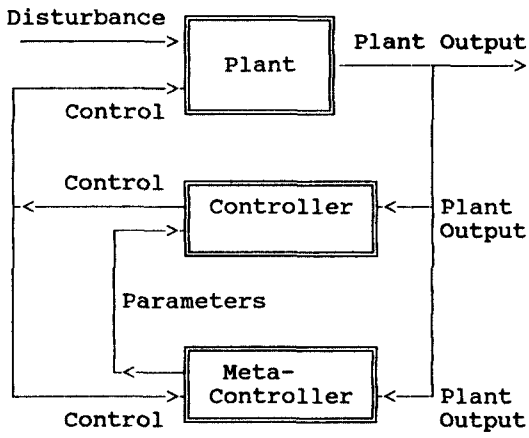

Table 1: Initial Spreadsheet Tableau

	INPUT TO SIMULATOR			OUTPUT FROM SIMULATOR		
	0.582 newθ	0.954 newθ'	3.606 newΓ	newT	newI	Better
	θ	θ'	Γ	time left	insta-bility	
1	0.03	0.2	1	0	83.2758	
2	0.03	0.2	10	0	100.012	
3	0.03	2	1	30	569	
4	1	0.2	10	154	123000	
5	1	0.2	1	213	2570000	
6	0.03	2	10	219	723000	
7	1	2	1	223	792000	
8	2	2	2	223	4600000	
9	1	2	10	223	5090000	
	badθ	badθ'	badΓ	badT	badI	
	0.64	1.1	4.5	centroid		
	-0.36	-0.9	-5.5	vector		0.1625 coefficient

Table 2: Final Spreadsheet Tableau

	INPUT TO SIMULATOR			OUTPUT FROM SIMULATOR		
	0.028 newθ	0.485 newθ'	6.601 newΓ	newT	newI	Better
	θ	θ'	Γ	time left	insta-bility	
1	0.023	0.347	7.243	0	12.2303	
2	0.027	0.435	6.034	0	22.6592	
3	0.031	0.62	6.522	0	31.7899	
4	0.031	0.515	4.781	0	35.0634	
5	0.027	0.444	6.598	0	39.342	
6	0.032	0.59	7.159	0	39.9405	
7	0.028	0.578	9.925	0	46.3209	
8	0.022	0.333	4.546	0	46.7025	
9	0.021	0.116	6.561	0	49.9	
	badθ	badθ'	badΓ	badT	badI	
	0.027	0.485	6.601	centroid		
	0.007	0.369	0.04	vector		0.0013 coefficient