Optimal boundary control of a tracking problem for a parabolic distributed system using hierarchical fuzzy control and evolutionary algorithms

R.J. Stonier, M.J. Drumm and J. Bell
Faculty of Informatics and Communication
Central Queensland University
Rockhampton, Queensland 4702, Australia
r.stonier@cqu.edu.au, m.drumm@cqu.edu.au, j.bell@cqu.edu.au

Abstract

In this paper we examine the application of hierarchical fuzzy control and evolutionary algorithms to find closed-loop control solutions of the optimal control problem arising from semi-discretisation of a linear parabolic tracking problem with boundary control. The solution is compared with that obtained by closed-loop methods based upon the variational equations of the Minimum Principle and the finite element method.

1 Introduction

A comparative study was made of five methods for calculating the optimal control function for a linear parabolic tracking problem with boundary control in Huntley [1]. Both open-loop methods based upon the variational equations and closed-loop methods via the Ricatti equation, were analysed for computational efficiency, accuracy, ease of programming and robustness.

This boundary control problem whose underlying state equations were parabolic partial differential equations, was first converted to a classical optimal control problem with ordinary differential state constraints through a method of semi-discretisation with respect to the state variable (the method of lines), cf [1]. These problems have usually been solved using fully discretised difference or finite element methods in recent times. Yet it has been suggested [2], that semi-discrete approaches obtained with current methods of solving stiff systems of ordinary differential equations might have advantages.

Stonier et al. [3] showed that it was feasible to apply an evolutionary algorithm with simple operators [4], to learn an open-loop controller for this optimal control problem in the case of a constant target function in the state space.

For this boundary control problem, we examine the evolutionary learning of a closed-loop controller in the form of a five (5) layered hierarchical fuzzy system comparing our results again with those given in [1] and the finite element method.

The new formulation of the boundary control problem using semi-discretisation brings with it associated problems in solution, one being the ‘curse of dimensionality’ when such discretisation is made in state variables, as well as the discretisation in time when integrating the state equations.

2 Boundary control of a distributed process

We consider the parabolic boundary control problem from [1], described by the following equations:

\[
\frac{\partial x}{\partial t} = \frac{\partial^2 x}{\partial y^2}, \quad 0 < t < T, \quad 0 < y < L,
\]

subject to boundary conditions

\[
x(y, 0) = 0
\]

\[
\begin{align*}
\frac{\partial x}{\partial y} &= \rho(x - u) \quad \text{on } y = 0 \\
\frac{\partial x}{\partial y} &= 0 \quad \text{on } y = L
\end{align*}
\]

0 < t ≤ T.

where \(\rho\) is a constant heat-transfer coefficient, \(x\) is temperature, \(y\) is depth and \(t\) is time. The process of semi-discretisation may be described by replacing
\[ \frac{\partial^2 x}{\partial y^2} \text{ by its central difference approximation} \]
\[ \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2} \text{ of local accuracy } O(h^2), \]
using \[ \Delta y = L/(N - 1) = h, \]
\[ \left. \frac{\partial x}{\partial y} \right|_{y=0} \text{ by } (x_1 - x_{-1})/2h, \]
\[ \left. \frac{\partial x}{\partial y} \right|_{y=L} \text{ by } (x_{N+1} - x_{N-1})/2h, \]
Defining \( x = [x_0, \cdots, x_N]^T \), the equations are discretised to
\[ \dot{x} = Ax + Bu, \quad (1) \]
where \( u \) is the scalar boundary control function, \( A \) is constant tri-diagonal matrix, and \( B \) is the vector \([2/\rho h, 0, \cdots, 0]^T\). The general analysis for the constant coefficient parabolic problem in one spatial dimension with boundary conditions is given in \([1]\).

For this problem the cost function defined by
\[ J = \frac{1}{2} \int_0^L [x(y, T) - \eta(y, t)]^2 dy + \frac{1}{2} \int_0^T ru^2(t) dt \]
where \( \eta(y, T) \) for \( 0 < y < L \), is the target function specified at all depths \( y \) for the dependent variable \( x \), to be minimised subject to the differential constraint given by Equation (1). The first term \( J_1 \) defines a quadratic performance measure of the error from the target profile, and the term \( J_2 \) a measure of cost in control.

The optimal control formulation now lies within the general formulation:
\[ \text{Minimise with respect to } u, \text{ the performance index} \]
\[ I[x, t_1] = \phi(x(t_1), t_1) + \int_{t_0}^{t_1} f_0(x(t), u(t), t) dt, \quad (3) \]
subject to the differential constraint
\[ \frac{dx}{dt} = f(x, u, t), \quad (4) \]
where the initial state \( x(t_0) \) is given and state vector \( x \in \mathbb{R}^n \), and control vector \( u \in \mathbb{R}^m \). The term \( \phi(x(t_1), t_1) \) usually represents a cost or penalty associated with the state at the final time \( t_1 \).

In general the system may be subject to combined state and control, equality and inequality constraints, and integral constraints as well as interior point constraints.

Mathematical programming, differential dynamic programming and gradient descent methods for this problem all typically require some form of conversion of the control function \( u \) into an approximately equivalent representation that consists of weighted combination/amalgamation of simpler functions (collocation methods). The problem then becomes one of finding optimal weights. Alternatively we can segment the continuous control functions by partitioning \([t_0, t_1]\) into \( NT \) intervals and replacing the control functions with simpler ones such as piecewise control for each \( u_t \) in the intervals \([t_i, t_{i+1}]\). The objective is then to find approximations in these local regions to optimise the performance integral. Such computation techniques lead to open-loop control solutions to the given problem. Evolutionary techniques for solving these problems can be found in \([5]\).

Although the problem is a simple one by definition, to find its solution is otherwise, for the stability of numerical approaches is a complicated function of \( \rho, r, \) number of state steps (in \( y \)), and the number of time points (in \( t \)). The mathematical analysis of finding an closed-loop control of this system using necessary conditions arising from the application the Minimum Principle is given in \([1]\). Two methods were used called Method 1 and Method 2. In Method 1 a standard fourth-order Runge-Kutta algorithm was used for the simultaneous backward integration of the associated matrix Riccati equation and costate equations. These were stored and used in the forward integration of the state equations using the same Runge-Kutta algorithm. In Method 2, the Kalman-Englar algorithm was used for the Riccati matrix alone and for the other integrations the Padé rational approximation method was used in conjunction with the Crank-Nicholson implicit method.

Our discussion concerns the case of a constant target function \( \eta = 0.2 \) using the parameters defined in \([1]\): \( L = 1, N = 10, T = 0.4, \rho = 1.0 \) and parameter \( r = 0.02 \).

3 Hierarchical fuzzy control

The total number of rules in a single layered fuzzy system is known to be an exponential function of the number of system variables, \([6]\). In this problem given that that \( N = 10 \), there are 11 component variables of the state vector. If each of these was identified by 3 fuzzy membership functions there would be \( 3^{11} \) rules to be determined. Clearly an unrealisable task. This ‘curse of dimensionality’ can be overcome by forming a hierarchical fuzzy control structure, one in which the most influential parameters are chosen as the system
variables in the first level, the next most important parameters are chosen as system variables in the second level, and so on, [7]. The first level of this hierarchy gives an approximate output which is then modified by the second level rule set. The procedure can be repeated in succeeding levels of hierarchy. In can be shown that in this hierarchical structure the number of rules in a complete rule base is reduced to a linear function of the total number of variables. Consider the decomposition of our problem in a five (5) layered hierarchical system as shown in Figure 1. Suppose that the output $u$ which is updated as output from each layer, can be described with 4 input membership functions to the next layer, and each of the remaining input variables can be as above, described using 3 membership functions. Then the total number of rules to be learnt is simply $3^3 + 4(4(3)^2) = 171$, a substantial reduction. We note the decomposition for this hierarchical structure is not unique. Input membership functions used in this paper are given in the following two figures, the first is the same for each component $x_k$ of the state vector, and the second for the output control $u$ that is updated at each layer.

Consider the first layer. There are $27 = 3^3$ rules in the knowledge base. In general, we may write the $\ell$th fuzzy rule has the form:

If $(x_0$ is $A^\ell_0$) and $(x_1$ is $A^\ell_1$) and $(x_2$ is $A^\ell_2$) Then $(u$ is $B^\ell)$.

where $A^\ell_k, k = 0, 1, 2$ are normalised fuzzy sets for input variables $x_k, k = 0, 1, 2$, respectively, and where $B^\ell$ are normalised fuzzy sets for output variable $u$.

For the second layer, and similarly for subsequent layers, there are $36 = 4(3)^3$ rules in the knowledge base and we may may write the $\ell$th fuzzy rule has the form:

If $(u$ is $C^\ell$) and $(x_3$ is $A^\ell_3$) and $(x_4$ is $A^\ell_2$) Then $(u$ is $B^\ell)$.

where $C^\ell$ are normalised fuzzy sets for the input control variable $U$.

Given a fuzzy rule base with $M$ rules, a fuzzy controller as given in Equation 5 uses a singleton fuzzifier, Mamdani product inference engine and centre average defuzzifier to determine output variables.

$$u = \frac{\sum_{\ell=1}^{M} \mu^{\ell} \left( \prod_{i=1}^{n} \mu^{\ell}_{A^\ell_i}(x_i) \right)}{\sum_{\ell=1}^{M} \left( \prod_{i=1}^{n} \mu^{\ell}_{A^\ell_i}(x_i) \right)}$$

(5)

where $\mu^{\ell}$ are centres of the output sets $B^\ell$.

In this paper we shall use 6 centres associated with the output sets $B^\ell$. They are equally spaced on the
interval $[-1,2]$, including both end points. This interval has been chosen from information on the control bounds given in [1].

The reduced number of fuzzy rules in this hierarchical structure can be learnt using an evolutionary algorithm, see for example, in [8]. Some theoretical aspects to be considered when designing HFL systems can be found in [7].

4 Solution by evolutionary algorithm

In this section we show how to apply an EA [4], to learn the fuzzy rules in the five knowledge bases. Each individual string in the evolutionary population is to uniquely represent the hierarchical structure. This is achieved as follows. In the knowledge base of any layer each fuzzy rule is uniquely defined by the consequent part, it being represented by an integer $\alpha_k \in [1,6]$ since there are six unknown centres in the formulation. In a similar manner each fuzzy rule in the second knowledge base is defined by its consequent.

The five fuzzy rule bases can therefore be represented as a linear individual string of $M = 27 + 4(36) = 171$ consequents,

$$u_k = [a_1, \cdots, a_{171}],$$

where $a_j$ is an integer $\in [1,6]$ for $j = 1, \cdots, 171$.

The initial population $P(0) = \{u_k : k = 1, \cdots, M\}$, where $M = 100$ is the number of strings, the size of the evolutionary population, was determined by choosing the $a_j$ as a random integer in $\{1, \cdots, 6\}$.

In determining successive populations a full replacement policy was used, tournament selection with size $n_T = 4$ and a modified mutation operator. An elitism policy was also used with two (2) copies of the best string from a given generation passed to the next generation.

Mutation (with probability $p_m = 0.01$), was defined by the following pseudo code:

```plaintext
if (mutate) {
    if (ak = 6) ak = ak - rnd(1,3);
    else if (ak = 1) ak = ak + rnd(1,3);
    else if (flip(0.5)) ak = ak + rnd(1,3);
    else ak = ak - rnd(1,3);
    if (ak > 6) ak = 6;
    if (ak < 1) ak = 1;
}
```

Crossover of parent strings to form two children in the next generation was taken as the usual one-point crossover with $p_c = 0.6$.

The fitness (objective) function for each string was determined as

$$F_{\text{obj}} = A(J_1) + A(J_2) + P_1 + P_2 + P_3,$$

where $A(J_k), k = 1, 2$ was a Trapezoidal approximation to the integrals $J_k$, and $P_k, k = 1, 2, 3$ are penalty terms defined below:

$$P_1 = \alpha_1 u_{201}^2, \quad P_2 = \alpha_2 \sum_{k=1}^{200} (u_{k+1} - u_k)^2,$$

$$P_3 = \alpha_3 \sum_{k=2}^{200} (u_k - u_{k-1})(u_{k+1} - u_k)$$

if $(u_k - u_{k-1})(u_{k+1} - u_k) < 0$.

The first penalty term was introduced to force the final value of the control to be zero at the final time $T$, a necessary requirement of the minimum principle. The second penalty seeks to ensure the sum of the squares of the differences in the piecewise constant approximation to the control remains small and the third seeks to remove any spiking in the graph due to gradient changes under mutation. Without the use of penalties such as these the solution found by the evolutionary algorithm tended to be ‘bang-bang’. For the given set of parameters the following values of $\alpha$ were typical: $\alpha_1 = 1, \alpha_2 = 10^{-2}$ and $\alpha_3 = 10^{-5}$.

To integrate the state equations (1) using a Runge-Kutta algorithm the time interval was discretised into $N_T = 401$ fixed time steps.

Constant Target Case $r = 0.02$

Our initial discussion concerns the case of a constant target function $\eta = 0.2$ using the parameters defined in [1]: $L = 1, T = 0.4, r = 0.02$, and $\rho = 1.0$.

The results we obtained using the evolutionary algorithm are compared with the Finite Element method and Methods 1 and 2 in the following table. The algorithm was terminated at generation 770. The value of the performance index $J_1 + J_2$ obtained was approximately 0.002779 for the evolutionary algorithm and 0.002766 for the finite element method. Given numerous runs of the evolutionary algorithm with different initial populations, it was found the algorithm consistently converged to a performance index approximately 0.00277 within 2000 generations.

The control graphs obtained for the Finite Element Method and Evolutionary algorithm shown in Figure 5 are in excellent agreement.
5 Conclusion

We have shown how to find a closed loop control in the form of a hierarchical fuzzy system for the boundary control of a distributed system converted to an optimal control problem using semi-discritisation.

It was found necessary to construct a hierarchical fuzzy controller with a number of layers in order to reduce the number of rules to a number. These fuzzy rules were found using an evolutionary algorithm. The set of rules learnt is not necessarily unique but they are shown to provide adequate accuracy when compared with other methods applied to this problem. In this structure the input membership functions for each updated \( u \) was obtained by trial and error, examining the output from each layer until a suitable range of values was determined. The layered structure is also not unique. A different number of layers could have been used and also a different arrangement of input to each layer.

The results are shown only for the constant target case and for the case \( r = 0.02 \). They were found to be in excellent agreement with the closed techniques applied in [1] and with the finite element. Successful application has also been made with other values of the parameter \( r \) and for the other two target functions described in [1] but are not presented.

References