Optimal boundary control of a tracking problem for a parabolic distributed system with open-loop control using evolutionary algorithms

Russel J STONIER
Faculty of Informatics and Communication,
Central Queensland University,
Rockhampton, QLD 4702, Australia

and

Michael J DRUMM
Faculty of Informatics and Communication,
Central Queensland University,
Rockhampton, QLD 4702, Australia

ABSTRACT

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

Keywords: Open-loop control, Optimal control, Boundary control, Evolutionary algorithms, Parabolic distributed system.

1. INTRODUCTION

In Huntley [1] a comparative study was made of five methods for calculating the optimal control function for a linear parabolic tracking problem with boundary control. 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]. In recent years it has been the practice to tackle these problems using fully discretised difference or finite element methods. 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, 5], to learn an open-loop controller for the classical optimal control problem in the case of a constant target function in the state variable.

In this paper we first show that for the constant target function better results can be obtained than those presented in [3] for the evolutionary learning of an open-loop control by simple changes in the mutation and crossover operator.

Huntley [1], also studied two other target functions, one a flat top ramp function and the other triangular as a result of the singularity arising at one ends in the constant case. We present here the evolutionary learning of an open-loop controller in each of these cases, comparing our results with those given in [1] and the finite element method. Penalties are included in the fitness evaluation of each string to ensure end compliance with the Minimum Principle and smoothness of the control solution in time.

The new formulation of the boundary control problem using semi-discretisation brings with it associated problems in solution, one being the‘curse of dimen-

sionality' when such discretisation is made in state variables, as well as the discretisation in time when solving the optimal control for a piecewise constant control strategy.

2. CLASSICAL OPTIMAL CONTROL

The *optimal control problem* stated briefly is:

Minimise with respect to $\underset{\sim}{u}$, the performance index

$$I[\underset{\sim}{x}, t_1] = \phi(\underset{\sim}{x}(t_1), t_1) + \int_{t_0}^{t_1} f_0(\underset{\sim}{x}(t), \underset{\sim}{u}(t), t)dt, \quad (1)$$

subject to the differential constraint

$$\frac{d\underset{\sim}{x}}{dt} = \mathbf{f}(\underset{\sim}{x}, \underset{\sim}{u}, t), \quad (2)$$

where the initial state $\underset{\sim}{x}(t_0)$ is given and state vector $\underset{\sim}{x} \in R^n$, and control vector $\underset{\sim}{u} \in R^m$. The term $\phi(\underset{\sim}{x}(t_1), t_1)$ 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 programing and gradient descent methods for this problem all typically require some form of conversion of the control function $\underset{\sim}{u}$ 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 N intervals and replacing the control functions with simpler ones such as piecewise control for each u_i in the intervals $[t_i, t_{i+1}]$. The objective is then to find approximations in these local regions to optimise the performance integral.

To apply an evolutionary algorithm we need a representation of a control strategy over $[t_0, t_1]$ which can be easily manipulated by operators that mimic and enhance the classical genetic operators of crossover and mutation. One approach is to consider each real encoded string in the population as an array of N m-vectors if the control is a piecewise constant approximation locally, (the approach taken in this paper for application in the open-loop case) or double m-vectors if the approximation is piece-wise linear.

Mutation or perturbation of the string can be done at the local or global level. Clearly if all strings have the same number of time partitions then a typical arithmetic crossover will, if the control space is convex, guarantee a valid string in the population. One-point crossover is more difficult to implement, cf [6].

Constraints are a known major problem in *optimal control* problems for we are solving a *functional* optimisation problem in which the solution $\underset{\sim}{u}$ belongs in control function space and NOT in the traditional state space. The differential constraints must be integrated to enable testing of interior state constraints and any equality or inequality constraints involving the state variables.

To achieve good convergence to a solution of these problems, as well as real life industrial optimisation problems, requires a new generation of genetic operators when there are a large number of real-valued parameters which have to be optimised in the presence of *equality and inequality* constraints, see for example, [6, 7].

3. 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

$$x(y, 0) = 0$$

$$\left.\begin{array}{rcll} \dfrac{\partial x}{\partial y} & = & \rho(x-u) & \text{on } y = 0 \\ \dfrac{\partial x}{\partial y} & = & 0 & \text{on } y = L \end{array}\right\} \quad 0 < t \leq T.$$

where ρ 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

- $\dfrac{\partial^2 x}{\partial y^2}$ by its central difference approximation $\dfrac{x_{i+1} - 2x_i + x_{i-1}}{h^2}$ of local accuracy $O(h^2)$, using $\Delta y = L/(N-1) = h$,
- $\left[\dfrac{\partial x}{\partial y}\right]_{y=0}$ by $(x_1 - x_{-1})/2h$, and $\left[\dfrac{\partial x}{\partial y}\right]_{y=L}$ by $(x_{N+1} - x_{N-1})/2h$,

Defining $\underset{\sim}{x} = [x_0, \cdots, x_N]^T$, the equations are discretised to

$$\dot{\underset{\sim}{x}} = A\underset{\sim}{x} + B, \tag{3}$$

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 to be minimised is defined by

$$\begin{aligned} J &= \frac{1}{2}\int_0^L [x(y,T) - \eta(y,t)]^2 \, dy + \frac{1}{2}\int_0^T ru^2(t) \, dt \\ &= J_1 + J_2, \end{aligned} \tag{4}$$

where $\eta(y,T)$ for $0 < y < L$, is the target function specified at all depths y for the dependent variable x. 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.

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 ρ, r, number of state steps (in y), and the number of time points (in t). The mathematical analysis of finding an *open-loop* control of this system using necessary conditions arising from the application the Minimum Principle is also given in [1].

4. SOLUTION BY EVOLUTIONARY ALGORITHM

To integrate the state equations (3) using a Runge-Kutta algorithm the time interval was discretised into $NT = 100$ *fixed* time steps. As each subinterval $[t_i, t_{i+1}]$ the Runge-Kutta algorithm requires a control evaluation at the midpoint of the interval, we construct a possible solution string in the population for the desired optimising control, as an array $\underset{\sim}{u}$ of 200 constant real elements u_i.

The initial population $P(0) = \{\underset{\sim}{u}^k \ : \ k = 1, \cdots, M\}$, where $M = 100$ is the number of strings, the size of the evolutionary population, was determined by choosing the u_i^k randomly in the interval $[-1, 2]$. The global upper bound $U_m = 2$ and global lower bound $U_\ell = -1$ were chosen from the control function graphs given in [1]. This defines a convex region in the control space.

In determining successive populations a full replacement policy was used, tournament selection with size $n_T = 4$ and a modified arithmetic crossover, see below, to determine two children in the next generation. An elitism policy was also used with four (4) copies of the best string from a given generation passed to the next generation.

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

$$F_{obj} = \beta_1 A(J_1) + \beta_2 A(J_2) + P_1 + P_2 + P_3,$$

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

$$\begin{aligned} P_1 &= \alpha_3 u_{201}^2, \qquad P_2 = \alpha_1 \sum_{k=1}^{200} (u_{k+1} - u_k)^2, \\ P_3 &= \alpha_2 \sum_{k=2}^{200} (u_k - u_{k-1})(u_{k+1} - u_k) \\ &\quad \text{if } \ (u_k - u_{k-1})(u_{k+1} - u_k) < 0. \end{aligned}$$

The constants $\beta_k, k = 1, 2$ are assumed positive and it is noted that the minimum of $\beta_1 A(J_1) + \beta_2 A(J_2)$ is the minimum of $A(J_1) + A(J_2)$ as both are positive. For the given set of parameters these constants were set at $\beta_1 = \beta_2 = 1$. A trapezoidal approximation was used for the integrals J_1 and J_2 in our calculations.

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 $\alpha_k, k = 1, 2$ were chosen small constants typically with value around 10^{-5} and 10^{-6} respectively, and $\alpha_3 = 1.0$.

In [3] mutation (with probability p_m), was taken in the following form approximating a small random Gaussian perturbation δu_k:

$$\begin{gathered} \text{If (mutate) } u_k = u_k + \delta u_k \text{ where} \\ \delta u_k = \alpha(\textstyle\sum_{k=1}^{12} r_k - 6)/6, \end{gathered}$$

where r_k was a random number in the interval $[0, 1]$, and α was a small constant factor, typically around 0.1. To ensure the perturbation realised feasible controls within the prescribed bounds, the new u_k was capped to lie in the desired interval, that is,

If $u_k > U_m$ then $u_k = U_m$, and
if $u_k < U_\ell$ then $u_k = U_\ell$.

Constant Target Case $r = 0.00001$
Our initial discussion concerns the case of a constant target function $\eta = 0.2$ using the parameters defined in [1]: $L = 1$, $T = 0.4$, $r = 0.00001$, and $\rho = 1.0$.

The results given in [3] were achieved at around generation 236400, with a population of 100, high mutation $p_m = 1.0$ for the first 50000 generations and then 0.7 thereafter, $p_c = 0.6$ and $\alpha = 0.06$ and in reasonable agreement with that obtained by the finite element method, see Figure 1. As can be seen, the control ob-

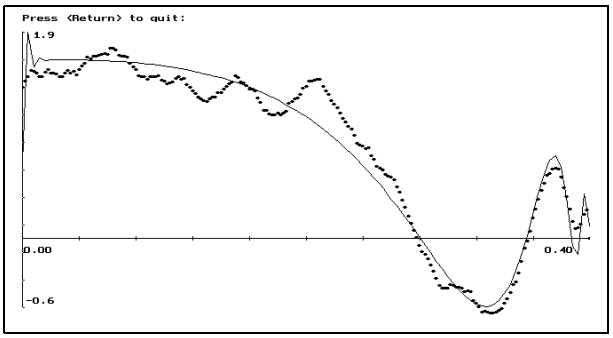

Figure 1: Control graphs for methods FE and EA

tained via the EA, does approximate well the solution obtained directly by the finite element method (the continuous line on the graph). The number of generations is unacceptably high and the EA depended critically on a high level of mutation and a small perturbation factor to enable tracking to the desired level of 0.2 to be effective.

In order to address these issues, let M(α) define the mutation given above, and take the new mutation operator to be:

```
if (mutate)
 if (flip(0.33))  M(1.0);
 else
    if (flip(0.5)) M(0.1);
    else
      if (flip(0.5))  pd = gen/maxgen;
         else  pd = 0.995;
      pow_ = ((1 - pd) * (1 - pd));
      fact = (1 - power(Random(), pow_));
      if (flip(0.5))
        delta = fact * (lb - u_k);
      else
        delta  = fact * (u - u_k);
      return (u_k + delta);
 else
   return(u_k);
```

If a control value u_k is mutated, then the basic mutation with a large factor $\alpha = 1$, is implemented one third of the time, basic mutation with a small factor $\alpha = 0.1$ one third of the time, and a modification of Michalewicz's mutation [5], is applied the remainder of the time. The standard arithmetic crossover with

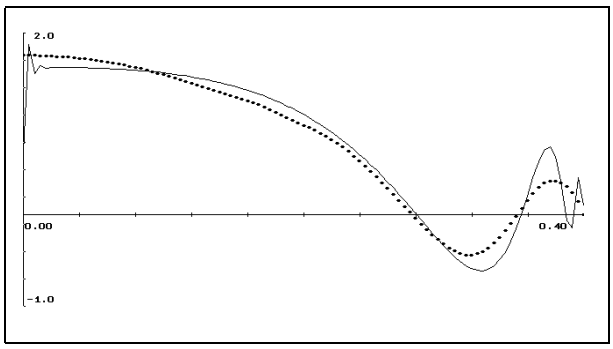

Figure 2: Control graphs for methods FE and EA with $r = 0.00001$

constant α_c and two parents generating two children was modified as follows: Given two parents $\underset{\sim}{p}_1$ and $\underset{\sim}{p}_2$ selected by tournament a child $\underset{\sim}{c}_1$ is then determined by random $\alpha_c \in [0, 1]$ from the equation

$$\underset{\sim}{c}_1 = \alpha_c \underset{\sim}{p}_1 + (1 - \alpha_c)\underset{\sim}{p}_2.$$

With continued selection of parent pairs, the children are added to the next generation's population until it is complete.

Applying these new operators with small mutation $p_m = 0.01$, the evolutionary algorithm yielded the control graph given by the dotted line in Figure 2 within 100000 generations. The results is a substantial improvement upon that obtained in [3]. Indeed the resultant control curve is very much as shown in 70000 generations.

Constant Target Case $r = 0.001$
Returning to the constant target case $\eta = 0.2$, with the new operators and with small $p_m = 0.01$, the evolutionary algorithm converged quickly within 7000 generations to the control graph shown by the dotted line in Figure 3. The value of $J_1 + J_2$ was 0.00021561 for the evolutionary algorithm and 0.0002143 for the finite element method.

We now discuss results for two other cases discussed in [1].

Ramp Case
In this case the target function is given by

$$\eta(y, T) = \begin{cases} 0.4(y/L) & 0 \le y/L < 0.5, \\ 0.2 & 0.5 \le y/L \le 1.0, \end{cases}$$

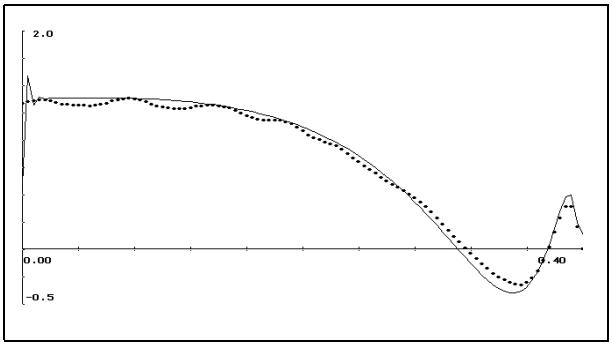

Figure 3: Control graphs for methods FE and EA in the constant case $r = 0.001$

Station y	Method 3	Method 4	FE	EA
0.0	0.2001	0.1953	0.2003	0.1991
0.1	0.1990	0.2004	0.1993	0.1989
0.2	0.1987	0.1980	0.1983	0.1992
0.3	0.2020	0.2020	0.2016	0.2023
0.4	0.2044	0.2040	0.2044	0.2046
0.5	0.2040	0.2041	0.2042	0.2039
0.6	0.2008	0.2009	0.2011	0.2006
0.7	0.1962	0.1962	0.1963	0.1960
0.8	0.1915	0.1918	0.1916	0.1914
0.9	0.1882	0.1882	0.1882	0.1881
1.0	0.1870	0.1874	0.1869	0.1869

Figure 4: Table 1 Comparison of output results

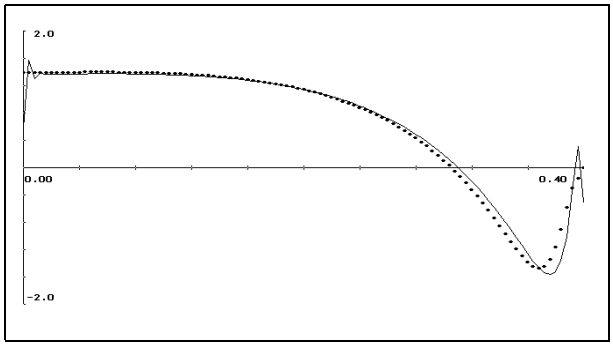

Figure 5: Control graphs for methods FE and EA in the ramp case

Station y	Ramp	Method 3	FE	EA
0.0	0.00	0.0011	0.0022	0.0028
0.1	0.04	0.0368	0.0367	0.0349
0.2	0.08	0.0846	0.0829	0.0857
0.3	0.12	0.1317	0.1316	0.1324
0.4	0.16	0.1658	0.1666	0.1656
0.5	0.20	0.1853	0.1862	0.1848
0.6	0.20	0.1939	0.1944	0.1935
0.7	0.20	0.1958	0.1959	0.1957
0.8	0.20	0.1947	0.1944	0.1947
0.9	0.20	0.1929	0.1925	0.1931
1.0	0.20	0.1922	0.1917	0.1924

Figure 6: Table 2 Comparison of output results

and the parameter r is maintained at the same value $r = 0.001$. This case is easier than the constant case as a non-zero target value at $y = 0$ is no longer demanded. In initialising the evolutionary algorithm and for the mutation operator, global bounds on the control were selected as $U_\ell = -7$ and $U_m = 6$.

Results obtained for a longer run of 300000 generations to obtain the control graph shown again by the dotted line in Figure 5. The approximation with the solution found by the finite element method is excellent. The value of $J_1 + J_2$ obtained was 0.00028373 for the evolutionary algorithm and 0.0002732 for the finite element method. A value of $J_1 + J_2$ of 0.000285 was obtained by the evolutionary algorithm within 11000 generations. Table 1 shows a comparision of the results of the evolutionary algorithm and the finite element method with Method 3 of [1] which is a numerical integration of the state and costate partial differential equations arising as necessary conditions from the minimum principle for distributed systems.

Triangle Case

The final case discussed by Huntley is that of a triangular target function with its peak at $y/L = 0.2$, defined by

$$\eta(y,T) = \begin{cases} 0.4(y/L) & 0 \le y/L < 0.5, \\ 0.4(L-y)/L & 0.5 \le y/L \le 1.0, \end{cases}$$

In this case we take the parameter $r = 0.0001$ and the global bounds on the control are taken to be those in the ramp case. Noting that boundary control is applied at one end of the spatial dimension only, it is not obvious that this difficult target function might be achieved computationally. Results shown in Figure 7 obtained for a run of 300000 generations to obtain the control graph shown again by the dotted line in Figure 5. The approximation with the solution found by the finite element method is excellent. The value of $J_1 + J_2$ obtained was 0.00033010 for the evolutionary algorithm and 0.0002249 for the finite element method.

These values target values are consistent with the graphical results for this case depicted in [1]. However the graphs of the control for both the evolutionary algorithm and finite element are not in good agreement. Further they are in disagreement with an extra oscil-

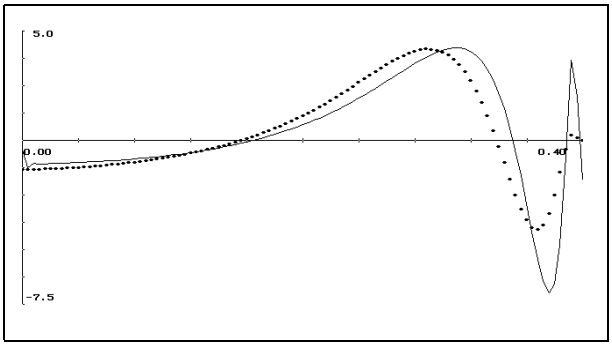

Figure 7: Control graphs for methods FE and EA in the triangle case

Station y	Triangle	FE	EA
0.0	0.00	-0.0009	0.0043
0.1	0.04	0.0423	0.0276
0.2	0.08	0.0701	0.0885
0.3	0.12	0.1358	0.1438
0.4	0.16	0.1763	0.1684
0.5	0.20	0.1742	0.1626
0.6	0.16	0.1450	0.1377
0.7	0.12	0.1076	0.1066
0.8	0.08	0.0748	0.0786
0.9	0.04	0.0532	0.0598
1.0	0.00	0.0458	0.0533

Figure 8: Table 3 Comparison of output results

lation in control at the later end of the time interval as described in Figure 5 of [1].

5. CONCLUSION

We have shown that evolutionary algorithms can be applied to optimal control problems in distributed parameter systems under semi-discretisaion with a good measure of success provided care is taken in correctly defining the right blend of operators and parameters.

Future research will continue to expand this application to the irrigation scheduling problem with the full water flow model to compare the results obtained with those obtained by the GA in [10].

A closed-loop control solution for the problem described above is under development to compare with the closed-loop methods given in [1]. It uses a hierarchical fuzzy controller with five layers.

6. REFERENCES

[1] E. Huntley, Optimal boundary control of a tracking problem for a parabolic distributed parameter system, *International Journal of Control*, Vol. 42, No. 2, 411-431, 1985.

[2] W.F. Ames, Numerical Methods for Partial Differential Equations, Academic Press, New York, 1977.

[3] R.J. Stonier et al., Optimal boundary control of a tracking problem using evolutionary algorithms, International Conference on Computational Intelligence for Modelling, Control and Automation, ISBN 0858898470, M. Mohammadian (Ed), Las Vegas, pp 204-214, 2001.

[4] T. Beck & H.P. Schwefel, Evolutionary Computation: An Overview, *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 20-29, 1996.

[5] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd Ed., Springer Verlag, 1994.

[6] S.F. Smith & R.J. Stonier, Applying evolutionary programming to constrained continuous optimal control problems, *Proceedings of IEEE International Conference on Evolutionary Computing*, Nagoya Japan, 285-290, 1996.

[7] S.F. Smith & R.J. Stonier, Evolutionary Operators for Constrained Continuous Optimal Control and High Dimensional Constrained Optimisation Problems, *Australian Journal of Intelligent Information Processing Systems*, Vol.4 No. 3/4, 240-248, 1997.

[8] S. Smith, The simplex method and evolutionary algorithms, *Proceedings of the International Conference on Evolutionary Computing*, Anchorage, Alaska, 799-804, 1998.

[9] R.J. Stonier, A. Stacey, M. Mohammadian & A.F. Smith, Applications of evolutionary learning in fuzzy logic and optimal control, *Computational Intelligence for Modelling ,Control and Automation, Evolutionary Computation, and Fuzzy Logic for Intelligent Control, Knowledge Acquisition and Information Retrieval*, M. Mohammadian (Ed), IOS Press, 76-85, 1999.

[10] R.J. Stonier & D.K. Sturgess, Genetic learning of the irrigation cycle for water flow in cropped soils, *Lecture Notes in Artificial Intelligence*, Xin Yao, Jong-Hwan Kim & Takeshi Furuhashi (Eds), Springer Verlag, Vol. 1285, 89-96, 1997.