BASIS FUNCTIONS, IDENTIFICATION AND GENETIC ALGORITHMS IN NORM-OPTIMAL ITERATIVE LEARNING CONTROL

V. Hatzikos[†], J. Hätönen[*,†], D. H. Owens[†]

[*] Systems Engineering Laboratory, University of Oulu, P.O.BOX 4300, FIN-90014 University of Oulu, Finland
[†] Department of Automatic Control and Systems Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK

Keywords: Iterative Learning Control, Genetic Algorithms, Optimal Control, System Identification

Abstract

Recently, in (Hatzikos and Owens, 2002*b*) and (Hatzikos and Owens, 2002*a*) it was explored whether or not Genetic Algorithm (GAs) based approach can be used in the context of norm-optimal Iterative Learning Control (ILC). It turned out the answer was positive for both linear and nonlinear plant models. However, this approach is still immature in the sense that it can produce very 'noisy' intermediate solutions. Furthermore, in practical applications the dimension of the search space can be very large, which can slow down considerably the GA algorithm and increase the computational burden. In order to overcome these problems, in this paper a new basis function approach is proposed. The idea is to restrict the GA search on a proper subspace of the original search space, where the subspace is spanned by a set of orthonormal functions. In this way it is possible to decrease the dimensionality of the search space, and if the basis functions are selected to be 'smooth', the search is done only over 'smooth' functions. It is in fact shown in this paper that under suitable assumptions, the basis function approach will result in monotonic convergence, which is a very strong property of an ILC algorithm. Furthermore, because the GA needs a simulation model of the plant in question, it is suggested in this paper that the input-output pairs from the ILC trials can be used to identify a model for the plant. Consequently this approach will result in an ILC algorithm with monotonic convergence that requires only an estimate of the order of the plant model. Simulations are used to illustrate the new approach, and they show that the basis function approach combined with identification gives good results in terms of convergence speed and input function smoothness.

1 Introduction

Iterative learning control is a technique to control systems operating in a repetitive mode with the additional requirement that a specified output trajectory $r(t)$ in an interval $[0, T]$ is to be followed with high precision. Examples of such systems are robot manipulators that are required to repeat a given task to high precision, chemical batch processes, or more generally, the class of tracking systems. It can be in fact stated that the repetitive processes comprise a very large group of industrial processes, ranging from robotics and semi-conductors to steels and chemical process industries. Motivated by human learning, the basic idea of iterative learning control is to use information from previous executions of the task in order to improve performance from trial to trial in the sense that the tracking error is sequentially reduced (Arimoto *et al.*, 1984), (Moore, 1993). (Arimoto *et al.*, 1984) in fact introduced a simple ILC algorithm that results in convergent learning, i.e. the tracking error goes to zero as the number of iterations increases.

Since the introduction of this convergent algorithm into the control community, several different algorithms have been suggested by several researchers, which will also result in convergent learning. However, most of the algorithms with guaranteed convergence properties work only for linear plants. Furthermore, they typically require that a fairly accurate model of the plant in question is available. These are severe limitations because the dynamics of a repetitive system can be highly nonlinear and it is typically very tedious build an accurate dynamical model for the real plant. Secondly, most of the process variables are subject to certain constraints that are set by safety considerations or physical constraints, resulting in a nonlinear problem setting. Hence there is a need for algorithms that can handle these hard constraints and nonlinear dynamics in a straightforward manner without an accurate mathematical model of the plant.

One straightforward approach is to use the optimality based algorithm presented in (Amann *et al.*, 1996), because even if the plant is nonlinear, this algorithm will result in monotonic convergence under suitable conditions. However, in most cases it is impossible to write down an analytic solution of the optimisation problem if the plant model in nonlinear. Hence in (Hatzikos and Owens, 2002*b*) it was suggested that the optimisation problem can be solved numerically by using Genetic Algorithms and a simulation model of the plant. The approach was shown to feasible with both linear and nonlinear examples. However, the algorithm had three drawbacks: in practical applications the dimensionality of the search space can be extremely large. Secondly, the GA tends to produce a very 'noisy' input function that cannot fed directly into a real plant (see (Hatzikos and Owens, 2002*a*). Finally, the algorithm needs an accurate plant model. Note that the noisy inputs can be avoided to a certain extent by filtering the input functions before feeding them into the real plant, but the design of the filter can very time-consuming. Hence the main theme of this paper is

to propose a basis function approach that will remove the first two undesirable properties of the GA-based approach. Furthermore, in this paper it is suggested that identification techniques can be used to build a local plant model from the trial data. This will result in an algorithm that requires only a rough estimate of the degree of the real plant.

The rest of the paper is organised as follows: In Section 2 a rigorous mathematical definition of the ILC problem is given. Section 3 explains the proposed basis function approach and shows under which conditions the resulting algorithm will convergent learning. Section 4 describes the identification method. Section 5 contains a detailed description of the algorithm implementation. Section 6 gives a numerical example, which shows the effectiveness of the proposed basis function approach combined with identification technique. Finally, Section 7 gives some conclusions and directions for future research.

2 ILC Problem definition

Consider the following possibly non-linear discrete-time dynamical system defined over finite time interval, $t \in [0, T_s, 2T_s \ \ldots, T_f]$:

$$\begin{aligned} x(t+Ts) &= f(x(t), u(t), t) \\ y(t) &= g(x(t), u(t), t) \end{aligned} \tag{1}$$

with a suitable initial condition $x(0) = x_0$. In addition, we are given a reference signal $r(t)$, and the control objective is make the output variable $y(t)$ to track this reference signal as closely as possible by manipulating the input variable $u(t)$. The special feature of the problem is that when the system (1) has reached the final time point $t = T_f$, the state of the system is reset back to x_0, and after the resetting the system is supposed to follow the same reference signal $r(t)$ again. This repetitive nature of the problem opens up possibilities for modifying iteratively the input function $u(t)$ so that as the number of repetitions or trials increases, the system learns the input function that gives perfect tracking. To be more precise, the idea is to find a control law

$$u_{k+1} = f(u_k, u_{k-1}, \ldots, u_{k-r}, e_{k+1}, e_k, \ldots, e_{k-s}) \tag{2}$$

so that

$$\lim_{k\to\infty} \|e_k\| \to 0 \quad \text{and} \quad \lim_{k\to\infty} \|u_k - u^*\| \to 0 \tag{3}$$

where u^* is the input function that gives perfect tracking (i.e. we are assuming the reference signal belongs to the range of the plant). Note that if the original plant model is a linear time-invariant model

$$\begin{aligned} x(t+T_s) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{4}$$

it can be represented equivalently with a matrix equation $y_k = G_e u_k$, where

$$G_e = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 \\ CB & 0 & 0 & \ldots & 0 \\ CAB & CB & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{T_l-1}B & CA^{T_l-2}B & \ldots & \ldots & 0 \end{bmatrix}, \tag{5}$$

where $T_l = T_f/T_s$, and $u_k = [u_k(0)\ u_k(T_s)\ \ldots\ u_k(T_f)]^T$, $y_k = [y_k(0)\ y_k(T_s)\ \ldots\ y_k(T_f)]^T$. This equivalent representation can typically simplify considerably the convergence analysis of ILC algorithms.

3 The basis function approach

In (Amann *et al.*, 1996) it was suggested that techniques from optimal control could be used to solve the ILC problem defined in the previous section. To be more precise, the idea is to solve the following optimisation problem

$$\min_{u_{k+1}} J(u_{k+1}) \tag{6}$$

$$J_{k+1}(u_{k+1}) = \|e_{k+1}\|^2 + \|u_{k+1} - u_k\|^2 \tag{7}$$

with the constraint equation $y_{k+1} = Gu_{k+1}$ between trial k and $k+1$, and the optimal input u^*_{k+1} is fed into the plant during trial $k+1$. G is the equivalent input-output mapping corresponding to (1). If the plant model G is linear, then the optimisation problem (6) can be solved analytically, and the resulting algorithm gives *geometric* convergence. However, if the plant model G is nonlinear, or constraints are imposed on the input function u_{k+1}, it might be very difficult to find an analytical solution for (6). To overcome this problem, in (Hatzikos and Owens, 2002*b*) it was propesed that Genetic Algorithms (GAs) could be used to search for the optimal u^*_{k+1} in (6) between trials using a simulation model of the plant G. The main idea behind this approach was to observe the following inequality holds based on (6) and (7)

$$\|e_{k+1}\|^2 \leq J_{k+1}(u^*_{k+1}) \leq \|e_k\|^2 \tag{8}$$

and hence if the optimisation problem (6) has at least one solution for $k = 1, 2, \ldots$, and the GA algorithm is able to find this solution, it holds that $\|e_{k+1}\| \leq \|e_k\|$, i.e. the approach results in monotonic convergence. Even it turned out that this method was feasible in the sense that it could produce almost zero tracking with both linear and nonlinear examples, the approach had two major drawbacks. The first drawback is that the because the GA is looking directly the optimal input function u^*_{k+1}, the dimension of the search space is going to be the length of trial, $T_l = T_f/T_s$, and in practical applications this number can be very large. Hence a mechanism is needed to reduce the dimension of the search space. Another drawback is the with this direct approach the GA will look for a vector u^*_{k+1} that gives good tracking, but it does not take into consideration whether or not the input function u^*_{k+1} is a smooth function. As was shown experimentally in (Hatzikos and Owens, 2002*b*)

and (Hatzikos and Owens, 2002*a*), this results in very 'noisy' input functions, that cannot necessarily fed directly into a real system. Hence another mechanism is needed to constrain the GA search over smooth functions.

One attractive way to achieve these two goals at the same time is to use a basis function approach. The idea is to write an input function u_{k+1} as

$$u_{k+1} = \sum_{i=1}^{T_l} \alpha_{i,k+1} f_i \tag{9}$$

where f_i is a set of orthonormal functions that span the input function space U. Furthermore, if we can assume that the first K basis functions span the optimising input function u^*_{k+1}, u^*_{k+1} can be written as $u^*_{k+1} = \sum_{i=1}^{K} \alpha^*_{i,k+1} f_i$ where $K < T_l$. Consequently we can now write the optimisation problem (6) equivalently in terms of the coefficients α_{k+1} in the following way:

$$\min_{\alpha_{k+1}} J_{k+1}(\alpha_{k+1}) \tag{10}$$

$$J_{k+1}(\alpha_{k+1}) = \|e_{k+1}\|^2 + \|\alpha_{k+1} - \alpha_k\|^2_{B_f^T B_f} \tag{11}$$

where B_f is the matrix $B_f := [f_1 \ f_2 \ \dots \ f_K]$. Note that $B_f^T B_f > 0$ by assumption, and hence the weighted norm $\|\cdot\|_{B_f^T B_f}$ is well-defined. Therefore by introducing the basis function, it is possible to reduce the dimension of the search space from T_l to K under the assumptions mentioned earlier. In addition, if the first K functions are smooth, their linear combination will be also smooth, and hence the GA will only look for smooth input functions u_{k+1}. It is also important to note that if the original constrain equation $y_{k+1} = Gu_{k+1}$ is linear, it can be written as $y_{k+1} = G_e u_{k+1} = G_e B_f \alpha_{k+1} = \tilde{G}_e \alpha_{k+1}$. Consequently if the original constraint equation is linear, the constraint equation in the basis function approach is linear as well, showing the optimisation problem does not become structurally more complex due the introduction of the basis functions. It is also easy to show that in the linear case the optimal control can be solved analytically and it is given by $\alpha^*_{k+1} = \alpha_k + \tilde{G}^*_e e_{k+1}$ where $\tilde{G}^*_e$ is the adjoint operator of $\tilde{G}_e$. Furthermore, in the linear case it is easy to show that the algorithm converges, and the limit of the sequence α_k satisfies

$$\tilde{G}^*_e \tilde{G}_e \alpha_\infty = \tilde{G}^*_e r, \tag{12}$$

which means that in the limit the algorithm solves the optimisation problem

$$\min_{\alpha} \|r - \tilde{G}_e \alpha\|^2. \tag{13}$$

Consequently if the input function u that gives perfect tracking does belongs to the span of the basis functions f_i, the algorithm will converge to that input function in the linear case. Furthermore, if u does not belong to the span of f_i, the algorithm looks for an optimal approximation of u.

The results in this section can be summarized with the following

Proposition 1 *Suppose that for $k = 1, 2, \dots$ the optimisation problem $\min_{u_{k+1}} J_{k+1}(u_{k+1})$ has at least one optimal solution u^*_{k+1}, for each k, $u^*_{k+1} \in$ Span$[f_1, \dots, f_K]$, and the GA is able to find $u^*_{k+1} \in$ Span$[f_1, \dots, f_K]$, then $\|e_{k+1}\| \leq \|e_k\|$.*

Proof. The proof is obvious based on the discussion above. □

The actual choice for the basis function set is still an open question. However, a natural candidate is the Fourier-basis

$$f_i(t) = \sin(i * \pi t / T_l) \tag{14}$$

for $i = 1, \dots, \ (K-1)/2$,

$$f_i(t) = \cos(i * \pi t / T_l) \tag{15}$$

for $k = (K-1)/2 + 1 \dots, \ K-1$, and $f_K = constant$. Another common basis function used in the context of dynamical systems is the Laguerre basis, defined by the recursive formula

$$(i+1) f_{i+1}(t) = (2i + 1 - t) f_i(t) - i f_{i-1}(t) \tag{16}$$

where the initial conditions are $f_1(t) = 1$ and $f_2(t) = -t + 1$. In Section 6 we will test how the Fourier-basis performs in the GA-ILC context.

4 Identification

In the previous work on GA-based ILC it was assumed that an accurate model of the plant to be controlled exists. However, this is not necessarily very feasible in applications, because the construction of a reasonably accurate plant model can be very time-consuming and tedious. In order to overcome this problem in this paper it is suggested that on-line identification could be used to construct a model from the experimental data available from the ILC trials. To be more precise, the plant model is parametrisised by using the model

$$A(z^{-1}) y(k) = B(z^{-1}) u(k) \tag{17}$$

where

$$\begin{aligned} A(z^{-1}) &= 1 + a_1 z^{-1} + \dots + a_n z^{-n} \\ B(z^{-1}) &= z^{-d}(b_1 z^{-1} + \dots + b_m z^{-m}) \end{aligned} \tag{18}$$

where $n, m \in \mathbb{N}_+$ and $m < n$ and the relative degree of the plant is $d - 1$. For this parametrisation it is a standard result from identification theory that for the input-ouput pair (u_{k+1}, y_{k+1}) the least-square estimate for the parameter vector $\beta := [-a_1 \ -a_2 \ \dots \ -a_n \ b_1 \ b_2 \ \dots \ b_n]$ is given by

$$\theta_k = (\Phi_k^T \Phi_k)^{-1} \Phi_k^T y_k \tag{19}$$

where

$$\Phi_k = \begin{bmatrix} \varphi_k^T(1) \\ \varphi_k^T(2) \\ \vdots \\ \varphi_k^T(N) \end{bmatrix} \tag{20}$$

and

$$\varphi_k(i) = \begin{bmatrix} -y(i-1) \\ \vdots \\ -y(i-n) \\ u(i-d-1) \\ \vdots \\ u(i-d-m) \end{bmatrix} \tag{21}$$

Note, however, that because the input u_k is given by an ILC algorithm, it is not necessarily true that the input will excite all the different modes of the plant, and hence the estimated parameters could be biased. The fact that the model is biased, is, however, not that severe as it sounds, because the algorithm the ILC needs only a local model of the plant. This is due the term $\|u_{k+1} - u_k\|^2$ in the cost function (7), which forces the optimal solution u_{k+1} be close to u_k. Hence if the model constructed from the input-output pair (u_k, y_k) gives a reasonable model for the input-output pairs in the vicinity of this operating point, it can be expected that the algorithm will converge.

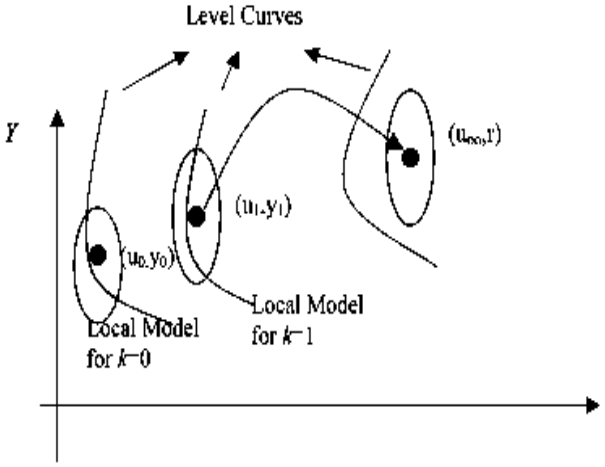

Figure 1: A flow diagram of the implementation.

This idea is illustated in Fig. 1: as a starting point an initial guess u_0 is fed into the real plant which gives an ouput function y_0. This data is used to constuct a least squares model for the real plant. After that the optimisation problem (6) is solved with this model, where the model is only accurate inside the ball in Fig. 1 centred at (u_0, y_0). However, because the cost function (7) includes the term $\|u_1 - u_0\|^2$, it can be argued if the radius of ball does not have to be excessively large in order to guarantee that the new optimal input u_1 lies inside this ball. Hence even with the local model the algorithm is able to find the optimal input for u_1. After the new optimal input is found, a new local model is constructed from the pair (u_1, y_1), and the optimisation is repeated. This process is then repeated until the algorithm converges.

5 Algorithm implementation

The proposed Basis Functions and Genetic Algorithm based optimisation method for Iterative Learning control systems (BFGA-ILC) is similar to the one described in (Hatzikos and Owens, 2002*b*) and (Hatzikos and Owens, 2002*a*). However, a significant difference is the use of real value representation of the individuals inside the GA instead of binary that was used before. The procedure starts by generating a population of K orthonormal functions and a matrix containing the coefficients. Using the linear combination of these two the BFGA algorithm evaluates an initial population of smooth input functions. Then the algorithm evaluates the fitness of each combination of coefficients. After that, the selection of the fittest combinations takes place. The coefficient matrix is then reproduced using the genetic operators (crossover and mutation). This means that fittest combinations of coefficients have better change of being chosen for reproduction. Finally, the developed offsprings are reinserted into the population replacing the old coefficients. This loop is repeated 100 times (generations) before selecting the next input to be introduced into the Simulink model. The algorithm ends when the error of the system is minimized to an optimal solution. Note that in this implementation it is straightforward to include hard constraints in the input variable $u(t)$: this is due the fact in the linear combination

$$u_t = \sum_i^K \alpha_i f_i(t), \tag{22}$$

the GA algorithm takes an input argument the ranges R_i of the decision variables α_i. Hence if we decide to use the Fourier basis in (14) and (15), and we specify that $|\alpha_i| < R$, then the maximum amplitude of $u(t)$ is going to be $K * R$, because the maximum (minimum) of $\cos(t)$ and $\sin(t)$ is 1 (-1). Consequently the proposed approach cope very easily with hard constraints on the input function $u(t)$.

Note that in order to guarantee monotonic convergence, the GA optimisation has to include a generation gap process. The idea behind the process is to use elitism, i.e. from the previous generation at least the best individual is inserted without modification into the next generation. The overall algorithm that combines the Genetic Algorithm and the identification routine can be described with the following steps:

1) Select an initial guess u_0 for the input and observe the corresponding output y_0 from the real plant. Calculate an estimate for the plant model with this data.

2) Create initial population of individuals

3) Evaluate objective function J for each individual and the correponding fitness level F using the identified simulation model from Step 1 during $k = 1$ and otherwise from Step 4 when $k > 1$.

4) Select fittest individual to be fed into the real model. Evaluate the performance of the selected input with the real plant. If the tracking accuracy is acceptable, terminate the

algorithm. Otherwise identify a new model equation from the experimental data. Replace existing simulation model by the new model.

5) Perform Genetic Operators: Select fittest individuals from the existing population. Perform crossover and mutation operators on selected individuals in order to create new offsprings. Then insert new offsprings into the population using an elitism strategy. Go back to step 3.

6 Simulation results

As a simulation example (this simulation example is taken from (Hamamoto and Sugye, 2001)) consider the following plant model

$$G(s) = \frac{s+8}{s^3+10s^2+30s+8} \tag{23}$$

where the system is defined over the time interval $t \in [0,3]$ with a sampling rate $T_s = 0.1$. The reference signal is given by the equation

$$r(t) = L^{-1}\left\{\frac{1}{s}\frac{1000}{s^4+40s^3+4000s+10000}\right\} \tag{24}$$

where $L^{-1}(\cdot)$ is the Laplace inverse-operator. The settings of the algorithm were done according the guidelines presented in (Chipperfield, 1996), see also Table 1. The range of the αs

SGA parameter	Setting
Population size of α	300
Total Generations	100
Number of iterations	6
Coding	Real-value representation
Selection	Low-level stochastic universal sampling routine
Recombination	Shuffle crossover with reduced surrogate, probability=0.9
Mutation	Value-flipping, random probability
Generation gap	0.98
Elitism	Best 12 chromosomes of previous population forward to next one
Number of Fourier basis	31

Table 1: Algorithm parameters for the linear case

were chosen to be $[-5,5]$, and hence the maximum amplitude of the input signal is going to be $31 \cdot 5 = 151$. The cost function was chosen to be

$$J(u_{k+1}) = \|e_{k+1}\|^2 + 0.01\|u_{k+1} - u_k\|^2, \tag{25}$$

In order to investigate the robustness of the approach against uncertainty in the degree of the plant model, the identification of the plant was done by using a second-order model

$$(1 + a_1 z^{-1} + a_2 z^{-2})y(k) = (b_1 z^{-1} + b_2 z^{-2})u(k) \tag{26}$$

Using the above settings the obtained results were very satisfactory. The proposed algorithm was able to produce the optimal Fourier coefficients after a few iterations, and the convergence is monotonic, see Fig. 2 even the degree of the identification model was different from the degree of the true plant. Furthermore, due the smoothness of the basis functions the algorithm is able to produce a smooth input function during each trial, see Fig. 3. Finally, Fig. 4 shows the parameter estimate for a_1 in (26) as a function of the iteration round. This figure shows how the parameter value changes from iteration to another because the identification routine is only able to construct a local model around the operating point (u_k, y_k).

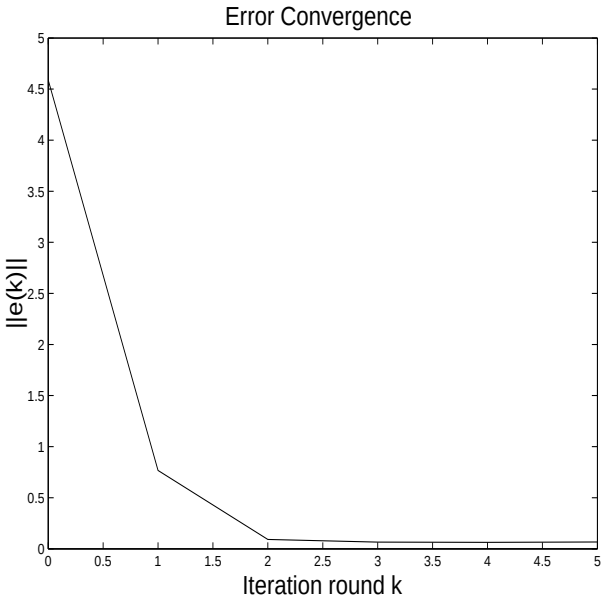

Figure 2: $\|\vec{e}(k)\|$ as a function of iteration round k with the linear example.

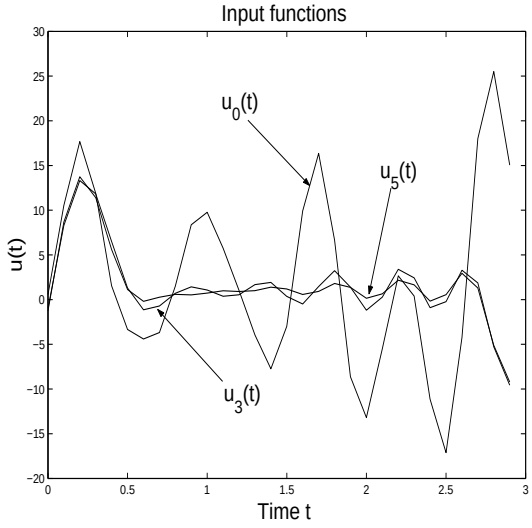

Figure 3: Input functions when $k = 0, 3, 5$.

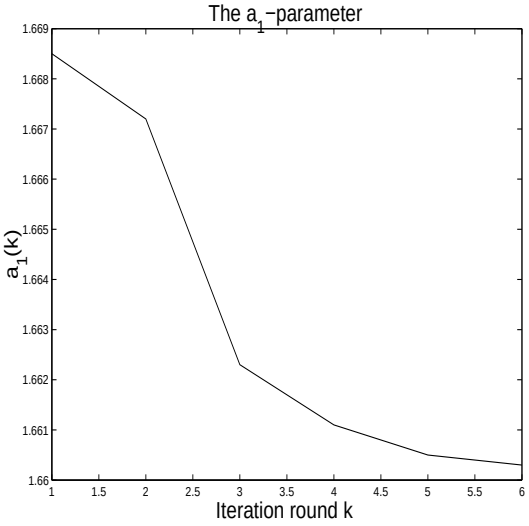

Figure 4: a_1 in the model (26) for $k = 0, 3, 5$.

7 Conclusions

In this paper the possibility of using GAs, identification and basis functions in the context of Norm-Optimal ILC algorithms were investigated. The basic idea behind the GA approach is that it can be used to implement Norm-Optimal ILC with linear and nonlinear plant models. The major improvement in the approach presented in this paper is that it should be computationally more effective than the previously presented approach in (Hatzikos and Owens, 2002*b*). In addition, the algorithm is capable of producing smooth input functions, whereas the algorithm in (Hatzikos and Owens, 2002*b*) typically results in very noisy input function during intermediate iterations, that need filtering before they can fed into the real plant. Furthermore, the new algorithm needs only a rough estimate on the degree of the plant, and it uses identification from previous data to build a identification model is used inside the GA algorithm.

This new approach was tried on a linear example, where the degree of the identification model was deliberately chosen to be different from the true plant model. The simulation experiment showed that in this case the algorithm produced only a local model around the operating point (u_k, y_k), but the algorithm still converged to the optimal input due the term $\|u_{k+1} - u_k\|^2$ in the cost function.

As a future work it should be investigated rigorously how accurate the local model has to be before convergence is achieved. In addition, it would be interesting to apply the scheme on nonlinear plant models.

8 Acknowledgements

J. Hätönen is supported by the EPSRC contract No GR/R74239/01.

References

Amann, N., D.H. Owens and E. Rogers (1996). Iterative learning control using optimal feedback and feedforward actions. *International Journal of Control* **65**(2), 277–293.

Arimoto, S., S. Kawamura and F. Miyazaki (1984). Bettering operations of robots by learning. *Journal of Robotic Systems* **1**, 123–140.

Chipperfield, A. (1996). Genetic algorithms toobox user's guide. Technical report. The University of Sheffield.

Hamamoto, K. and T. Sugye (2001). An iterative learning control algorithm within predescribed input-output subspace. *Automatica* **37**, 1803–1809.

Hatzikos, V. and D. H. Owens (2002*a*). A genetic algorithm based optimisation method for iterative learning control systems. In: *Proceedings of the 3th Workshop on Robot Motion and Control*. Poznan, Poland.

Hatzikos, V. and D. H. Owens (2002*b*). Introducing evolutionary based frameworks into iterative learning control applications. In: *The Proceedings of the 4th International Conference on Tecnology and Automation*. Thessaloniki, Greece.

Moore, K.L. (1993). *Iterative Learning Control for Deterministic Systems*. Springer-Verlag.