2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Performing Aggressive Maneuvers using Iterative Learning Control

Oliver Purwin
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, NY 14853, USA
op24@cornell.edu

Raffaello D'Andrea
Department of Mechanical and Process Engineering
ETH Zürich
Zürich, Switzerland
rdandrea@ethz.ch

Abstract— **This paper presents an algorithm to iteratively drive a system quickly from one state to another. A simple model which captures the essential features of the system is used to compute the reference trajectory as the solution of an optimal control problem. Based on a lifted domain description of that same model an iterative learning controller is synthesized by solving a linear least-squares problem. The non-causality of the approach makes it possible to anticipate recurring disturbances. Computational requirements are modest, allowing controller update in real-time. The experience gained from successful maneuvers can be used to significantly reduce transients when performing similar motions. The algorithm is successfully applied to a real quadrotor unmanned aerial vehicle. The results are presented and discussed.**

I. Introduction

With the increasing popularity of autonomous systems there arises a need to take advantage of their full capabilities. One approach to increase the performance is to identify the system well and apply advanced control methods. However, this possibly involves extensive system identification efforts. A different paradigm is to put the complexity in the software. A relatively simple model in conjunction with an adaptive algorithm and a well-chosen set of sensors allows each vehicle to experimentally determine how to perform a difficult maneuver and to compensate for individual differences in the system dynamics.

One data-based approach is called iterative learning control (ILC). The idea behind ILC is that the performance of a system executing the same kind of motion repeatedly can be improved by learning from previous executions. Given a desired output signal ILC algorithms experimentally determine an open-loop input signal which approximately inverts the system dynamics and yields the desired output. Bristow et al. [1] provide a survey of different design techniques for ILC.

Chen and Moore [2] present an approach based on local symmetrical double-integration of the feedback signal and apply it to a simulated omnidirectional ground vehicle. Ghosh and Paden [3] show an approach based on approximate inversion of the system dynamics. Chin et al. [4] merge a model predictive controller [5] with an ILC [6]. The real-time feedback component of this approach is intended to reject non-repetitive noise while the ILC adjusts to the repetitive disturbance. Cho et al. [7] put this approach in a state-space framework.

Non-causal control laws allow ILC to preemptively compensate for disturbances or model uncertainties which are constant from trial to trial. Formulating the problem and controller in the lifted domain is a natural way of exploiting the repetitive nature of the experiments, made viable by advancements in computer processors and memory. Rice and Verhaegen [8] present a structured unified approach to ILC synthesis based on the lifted state-space description of the plant/controller system.

In practice ILC have been applied to repetitive tasks performed by stationary systems, such as wafer stages [9], chemical reactors [10], or industrial robots [11]. Applications to autonomous vehicles are more rare.

This paper presents a lifted domain ILC algorithm which enables a system to perform an aggressive motion, i.e. drive the system from one state to another. Aggressive in this context characterizes a maneuver that takes place in the nonlinear regime of the system and/or close to the state or input constraints. This maneuver would be hard to tune by hand or would require very accurate knowledge of the underlying system. Instead, the featured algorithm only requires a comparatively simple model and initial guess for the input. In case of an unstable system it is assumed that a stabilizing controller is available.

The algorithm is intended for autonomous vehicles. However, it can be applied to a wide range of different dynamic systems without change. The controller update can be executed online with modest computational resources. If a particular maneuver is performed satisfactorily the gained knowledge can be utilized to perform a maneuver which is similar to the one just learned. This can reduce transients and improve convergence.

The first step of the algorithm is the computation of the reference trajectory and input by solving an optimal control problem. The nonlinear model is then linearized about the reference and discretized resulting in a discrete time linear time-varying (LTV) system. The lifted description of this LTV system defines the input-output relationship of the system for one complete experimental run in form of a single matrix. After performing an experiment the results are stored and compared with the ideal trajectory, yielding an error vector. Solving a linear least-squares (LLS) problem based on the lifted LTV system and the error vector yields the change in the input signal for the next trial. The algorithm terminates if the norm of the error vector is sufficiently small.

The algorithm is successfully applied to an unmanned aerial vehicle (UAV). After stabilization the UAV is a marginally stable system which requires accurate feedforward inputs to track a reference satisfactorily.

The rest of the paper is organized as follows: Section II presents the dynamics of the UAV used for algorithm derivation. Section III describes the algorithm to perform a single maneuver. The learned maneuver is extended in Section IV. Section V shows the successful application of the algorithm to the UAV, and Section VI provides a conclusion.

II. Vehicle Dynamics

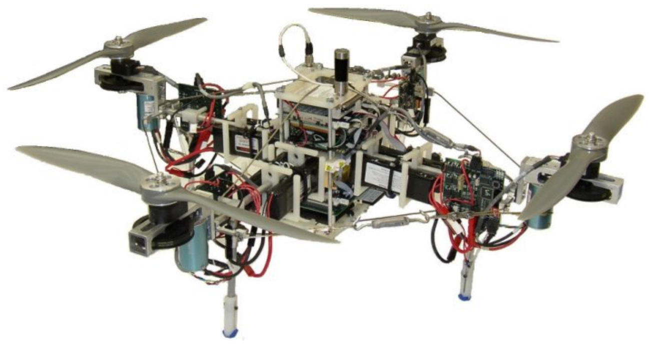
Fig. 1. Quadrotor Unmanned Aerial Vehicle

The presented algorithm is being applied to the quadrotor UAV shown in Figure 1. The dynamics of the vehicle are unstable so to perform experiments with the airborne vehicle an initial controller is required which stabilizes the UAV in hover before and after an ILC trial. A PD controller has been synthesized based on the linearized six degree of freedom (DOF) rigid body dynamics. In the following this controller will be referred to as hover controller.

The maneuver that will be covered in the subsequent sections of the paper consists of a sideways motion in the xz plane, see Figure 2 (left). This reduces the dynamics to two-dimensional space and three DOF without loss of generality. A motion in 3D involves a larger set of equations but is conceptually identical. Figure 2 (right) shows the free-body diagram of the equivalent 2D vehicle. The global frame of reference (FOR) is denoted by x, z, and θ. Thrust vectors f_0 and f_1 extend from the centers of the propellers and are offset from the vertical vehicle axis by θ_δ. The distances between thrust and center of mass are denoted by l_0 and l_1. Note that gravity g acts in positive z direction. The vehicle mass is denoted by m, the moment of inertia by j.

As mentioned before, any kind of more sophisticated dynamics such as aerodynamics or flexible structures are neglected for the sake of simplicity. Taking the force and moment balances yields the equations of motion of the rigid body in the global FOR:

$$\begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m}\sin(\theta+\theta_\delta)(f_0+f_1) \\ -\frac{1}{m}\cos(\theta+\theta_\delta)(f_0+f_1)+g \\ \frac{1}{j}(f_0 l_0 - f_1 l_1) \end{bmatrix} \quad (1)$$

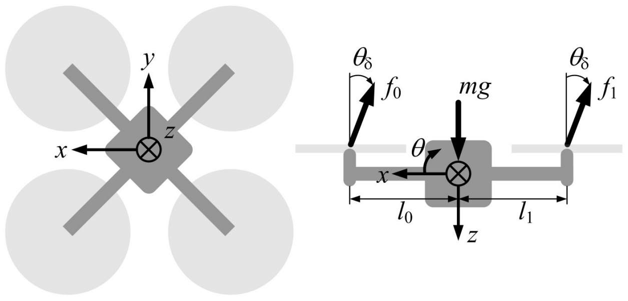

Fig. 2. UAV Top Down View (left), Free-body Diagram (right)

To stabilize the system and reduce the impact of nonlinearities a control loop is implemented for the rotation, such that the response of θ is that of a second order system

$$\begin{aligned} \ddot{\theta} &= -2\zeta\omega_n\dot{\theta} - \omega_n^2(\theta-\theta_c) && (2) \\ &= k_{\dot{\theta}}\dot{\theta} + k_\theta(\theta-\theta_c) && (3) \end{aligned}$$

with damping ratio ζ, natural frequency ω_n, and commanded angle θ_c. The parameters ζ and ω_n are selected to match the system response of the closed loop system with the hover controller in place. This guarantees stability and adequate performance of the θ stabilization in hover. Combining the thrust inputs f_0 and f_1 to

$$f_a = f_0 + f_1 \quad (4)$$

the stabilized equations of motion of the vehicle then become

$$\begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m}\sin(\theta+\theta_\delta)f_a \\ -\frac{1}{m}\cos(\theta+\theta_\delta)f_a+g \\ k_{\dot{\theta}}\dot{\theta} + k_\theta(\theta-\theta_c) \end{bmatrix} = \mathfrak{f}(\rho, q, u) \quad (5)$$

where f_a and θ_c are the new system inputs and the state $q(t)$ and the parameter vector ρ are defined as

$$\begin{aligned} q(t) &= \begin{bmatrix} x(t) & z(t) & \dot{x}(t) & \dot{z}(t) & \theta(t) & \dot{\theta}(t) \end{bmatrix}^T && (6) \\ \rho &= \begin{bmatrix} m & l_0 & l_1 & \theta_\delta \end{bmatrix}^T && (7) \end{aligned}$$

For specifics about the selection of the parameter vector see Section IV.

III. Performing a Maneuver

The objective of the presented algorithm is to perform a maneuver. For the purpose of this paper the initial state q_0 and target state q_f are both equilibria, i.e. the vehicle is hovering with $\dot{q} = 0$. Note that this is not a requirement of the algorithm. However, the advantages are better controlled initial conditions and increased safety, since the vehicle is not moving in q_0 and q_f and therefore not endangering itself or the environment. Performing a maneuver consists of two parts: generation of the reference trajectory $q_d(t)$ and learning how to track it.

A. Generation of a reference trajectory

The reference trajectory $q_d(t)$ is the solution of an optimal control problem (OCP): compute the minimum time solution

$$(q_d(t), u_{d,temp}(t)), \quad u_{d,temp}(t) = [f_{0,d}(t)\ f_{1,d}(t)]^T \tag{8}$$

which drives the system (1) from the initial state $q(0) = q_0$ to the final state $q(t_f) = q_f$, subject to constraints on the control effort

$$|f_i(t)| \leq f_{i,max}, \quad |\dot{f}_i(t)| \leq \dot{f}_{i,max} \tag{9}$$

with the maximum thrust $f_{i,max}$ and maximum rate of change of thrust $\dot{f}_{i,max}$ dependent on the used motor/battery/propeller combination. For the purpose of this paper the OCP is solved using RIOTS [12], an optimal control toolbox written in Matlab and C. The above formulation of the OCP has been chosen for simplicity in the expressions of the constraints, which benefits the numerical solution process of the OCP. However, in subsequent parts of the algorithm the inputs according to system (5) are being used. Therefore the optimal inputs $u_{d,temp}(t)$ have to be transformed to $u_d(t) = [f_{a,d}(t)\ \theta_{c,d}(t)]^T$, such that $u_d(t)$ applied to (5) yields $q_d(t)$. Substituting (3) into the third line of (1) yields

$$\frac{1}{j}(f_0 l_0 - f_1 l_1) \quad = \quad k_{\dot{\theta}}\dot{\theta} + k_\theta(\theta - \theta_c) \tag{10}$$

which can be solved for θ_c. Together with (4) this yields

$$f_{a,d}(t) \quad = \quad f_0(t) + f_1(t) \tag{11}$$

$$\theta_{c,d}(t) \quad = \quad \frac{k_{\dot{\theta}}\dot{\theta}_d(t) + k_\theta\theta_d(t) - \frac{1}{j}(f_0(t)l_0 - f_1(t)l_1)}{k_\theta} \tag{12}$$

Note that k_θ is not equal to zero if the control loop around θ has a proportional term, which is the case for the application at hand.

B. Tracking the reference trajectory

One basic assumption of the algorithm is that the motion of the vehicle stays close to the generated reference trajectory $q_d(t)$. Linearizing (5) about this trajectory and input yields

$$\dot{\tilde{q}}(t) \quad = \quad \left.\frac{\partial \mathfrak{f}}{\partial q}\right|_{q_d,u_d} \tilde{q}(t) + \left.\frac{\partial \mathfrak{f}}{\partial u}\right|_{q_d,u_d} \tilde{u}(t) \tag{13}$$

$$\quad = \quad A(t)\tilde{q}(t) + B(t)\tilde{u}(t) \tag{14}$$

with $q = q_d + \tilde{q}$ and $u = u_d + \tilde{u}$. Converting to a discrete time system results in a linear time-varying system

$$\tilde{q}(k+1) \quad = \quad A_D(k)\tilde{q}(k) + B_D(k)\tilde{u}(k) \tag{15}$$

with k denoting a discrete time step and N being the trial length. The dynamics (15) of a complete trial are written in the lifted domain to exploit the repetitiveness of the experiments and synthesize a non-causal controller

$$\tilde{Q} = P\tilde{U} \tag{16}$$

$$\tilde{Q} = \begin{bmatrix} \tilde{q}(1) \\ \tilde{q}(2) \\ \vdots \\ \tilde{q}(N) \end{bmatrix}, \quad \tilde{U} = \begin{bmatrix} \tilde{u}(0) \\ \tilde{u}(1) \\ \vdots \\ \tilde{u}(N-1) \end{bmatrix} \tag{17}$$

$$P = \begin{bmatrix} B_D(0) & 0 & \cdots & 0 \\ A_D(1)B_D(0) & B_D(1) & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \Phi B_D(0) & \cdots & & B_D(N-1) \end{bmatrix} \tag{18}$$

$$\Phi = A_D(N-1)A_D(N-2)...A_D(1) \tag{19}$$

where the capital letters $\tilde{Q}$ and $\tilde{U}$ indicate lifted versions of $\tilde{q}$ and $\tilde{u}$ while P denotes the matrix containing the lifted dynamics. A widely used [1] ILC approach is

$$U_{j+1} \quad = \quad L_1[U_j + L_2 E_j] \tag{20}$$

where the index j denotes the trial, L_1 and L_2 denote two filter functions (matrices), and E_j is the lifted error signal

$$E_j \quad = \quad Q_d - Q_{j,m}, \quad Q_{j,m} = Q_j + noise \tag{21}$$

with $Q_{j,m}$ representing noisy measurements. The presented ILC takes advantage of the given model which captures the essential dynamics of the underlying system. The exact formulation of the update law depends on the assumptions made about the noise. In case that the noise $d(k)$ does not change from trial to trial the system takes the form

$$q_{j,m}(k) \quad = \quad q_d(k) + \tilde{q}_j(k) + d(k) \tag{22}$$

$$Q_{j,m} \quad = \quad Q_d + \tilde{Q}_j + D \tag{23}$$

with D being the lifted constant disturbance vector, representing modeling errors or repeatable process noise for example. Using (16) and (21) it follows that

$$Q_{j,m} \quad = \quad Q_d + P\tilde{U}_j + D \tag{24}$$

$$E_j \quad = \quad Q_d - Q_{j,m} = -P\tilde{U}_j - D \tag{25}$$

Performing a single experiment or trial with $\tilde{U}_0 = 0$ yields an error signal of $E_0 = -D$. The input which minimizes the square of the error signal is the solution of a linear least-squares problem:

$$\tilde{U}_1 \quad = \quad \arg\min_{\tilde{U}} \|E\|_2^2 = \arg\min_{\tilde{U}} \|P\tilde{U} + D\|_2^2 \tag{26}$$

$$\quad = \quad -P^\dagger D \tag{27}$$

$$\quad = \quad \tilde{U}_0 + P^\dagger E_0 \tag{28}$$

$$\quad = \quad \tilde{U}_0 + L_2 E_0 \tag{29}$$

where $P^\dagger$ indicates the pseudo-inverse

$$P^\dagger \quad = \quad \lim_{\epsilon \to 0}(P^T P + \epsilon I)^{-1} P^T, \quad \epsilon > 0 \tag{30}$$

which can be computed by well-established methods such as singular value decomposition (SVD). The input $\tilde{U}_1$ applied to the same system (24), will result in an error of

$$E_1 \quad = \quad -(I - PP^\dagger)D \tag{31}$$

The update law $L_2 = P^\dagger$ can be non-causal, which results in a dense matrix $P^\dagger$. In case there is not only a constant disturbance but additional white noise $v_j(k)$ that changes from trial to trial the system takes the form

$$q_{j,m}(k) = q_d(k) + \tilde{q}_j(k) + d(k) + v_j(k) \quad (32)$$
$$Q_{j,m} = Q_d + \tilde{Q}_j + D + V_j \quad (33)$$
$$E_j = Q_d - Q_{j,m} = -P\tilde{U}_j - D - V_j \quad (34)$$

with V_j being the lifted noise. All components of $v_j(k)$ and V_j are assumed to be independent and identically distributed (iid) zero-mean Gaussian white noise. Using a similar approach as (28) for the update law results in

$$\tilde{U}_{j+1} = \tilde{U}_j + \alpha P^\dagger E_j, \quad \alpha \in (0,1) \quad (35)$$

Substituting (34) the input dynamics in the trial domain are

$$\tilde{U}_{j+1} = (I - \alpha P^\dagger P)\tilde{U}_j - \alpha P^\dagger D - \alpha P^\dagger V_j \quad (36)$$

which can readily be shown to be equal to

$$\tilde{U}_{j+1} = -\left[1 - (1-\alpha)^{j+1}\right] P^\dagger D - \alpha P^\dagger \sum_{i=0}^{j} (1-\alpha)^{j-i} V_i \quad (37)$$

while assuming $\tilde{U}_0 = 0$. In the limit with the number of iterations tending towards infinity the input and error become

$$\tilde{U}_\infty = -P^\dagger D - P^\dagger W \quad (38)$$
$$E_\infty = -[I - PP^\dagger]D - V_\infty + PP^\dagger W \quad (39)$$
$$W = \alpha \sum_{i=0}^{j} (1-\alpha)^{j-i} V_i \quad (40)$$

while introducing the equivalent noise vector W

$$\mathrm{E}[W] = 0, \quad \mathrm{E}[WW^T] = \frac{\alpha}{2-\alpha}\mathrm{E}[VV^T] \quad (41)$$

The parameter α serves as a tuning parameter to regulate the influence of V_j. For α approaching 1 the variance is not reduced, it is the same as performing the update only once. For α approaching zero the solution tends towards the optimum, i.e. minimizes the influence of V_j. The downside is that the number of iterations required to get this solution tends towards infinity. In practice the trade-off has to be somewhere in between.

In addition to the update law L_2 it is possible to introduce a low-pass filter in L_1, which rejects high frequency noise that gets injected by the measurements.

The algorithm terminates successfully if the error E_j is smaller than a specified threshold. In that case the final input is denoted $\tilde{U}_M$ and the final experimental trajectory is denoted $\tilde{Q}_M$.

IV. Extending the Maneuver

In the previous section an algorithm was presented which iteratively tracks a given trajectory. In this section a method is proposed which facilitates the tracking of $q_{d,2}(t)$, assuming that it is already known how to track $q_{d,1}(t)$, with $q_{d,1}(t)$ and $q_{d,2}(t)$ being similar. The most straightforward approach is the direct application of the algorithm from Section III. However, this would neglect valuable information gained from previous experiments. A better method is to utilize the final input $\tilde{U}_{M,1}$ and trajectory $\tilde{Q}_{M,1}$ from tracking $q_{d,1}(t)$ in order to provide better initial guesses for the tracking of $q_{d,2}(t)$. The approach described here involves the adjustment of the model parameters ρ (7). The goal is to adjust the nonlinear model such that the model (5) in conjunction with the real input explains the real output, i.e.

$$q_{M,1}(k) = q_{th,1}(k), \quad k \in [1, N] \quad (42)$$

with

$$q_{th,1}(k) = \int_0^{t(k)} \mathfrak{f}(\rho, q(\tau), u_{M,1}(\tau))d\tau, \quad q(0) = q_{M,1}(0) \quad (43)$$

Using lifted vectors this can be posed as a nonlinear quadratic optimization problem

$$\rho^* = \arg\min_\rho \|Q_{M,1} - Q_{th,1}\|_2^2 \quad (44)$$

This optimal ρ^* is then substituted into the model (5) to provide the basis for the algorithm as described in Section III.

The selection of adjustment parameters can be crucial for the convergence of an optimization process involving a nonlinear system. The particular vector (7) has been chosen since it allows the adjustment of the relationships between inputs and states. Further, it provides good results in practice, see Section V. However, it should be noted that this selection is not unique and that other parameter vectors could provide similar results.

The easiest approach is to define a ρ which is valid over the entire duration of the trial. However, to improve performance of the adjustment process it is possible to define N_ρ different parameter sets ρ_n which are valid during consecutive intervals $[k_{\rho,n,0}, k_{\rho,n,f}]$ of equal size Δk_ρ, i.e.

$$k_{\rho,n,f} + 1 = k_{\rho,n+1,0} \quad (45)$$
$$k_{\rho,n,f} - k_{\rho,n,0} = \Delta k_\rho \quad (46)$$

Figure 3 shows a plot of the residual of the optimization (44) over the number of parameter sets N_ρ for a typical experimental result ($\tilde{Q}_M$, $\tilde{U}_M$). For the actual experiments

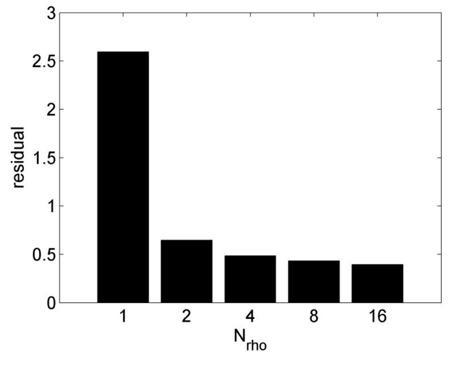

Fig. 3. Residual of Optimization over Number of Parameter Sets

N_ρ was set to four. The values of ρ_n^* are not unreasonably different from the unadjusted ρ_0, as can be seen in Table I.

TABLE I
COMPARISON OF ρ

Parameter	m [kg]	l_0 [m]	l_1 [m]	θ_δ [rad]
ρ_0	5.6	0.2623	0.2623	0
ρ_1^*	5.6318	0.2203	0.2233	-0.0054
ρ_2^*	5.4785	0.2059	0.1985	-0.0343
ρ_3^*	5.4744	0.2023	0.1973	0.1308
ρ_4^*	5.5770	0.1729	0.1649	0.0363

V. Implementation and Results

The algorithm was applied to the UAV shown in Figure 1. The iterative part of the ILC computations is implemented in C++ and executed while the vehicle is airborne. The computation of a single ILC iteration involves nonlinear transformations from 3D measurements to the 2D problem, low-pass filtering, and solving the LLS problem (26) by applying SVD. It takes about 5 to 10 seconds to complete one iteration on a 650 MHz Pentium processor. Therefore it is possible to perform all iterations necessary for successful termination of the algorithm during a single flight without having to land and recharge the batteries. The rest of the algorithm, such as solving the OCP and adjusting the parameter vector, is implemented in Matlab, since there is no need for online computation.

A side-to-side motion was selected as a characteristic trajectory to show the capabilities of the algorithm. Hovering at position q_0 the vehicle was required to move to position q_f as quickly as possible and come to a full stop. This motion involves a significant amount of nonlinearities due to the quick acceleration and deceleration phases. Further, the banking of the vehicle in combination with high linear velocities was likely to introduce complex fluid dynamic effects. These effects were not explicitly modeled but expected to be part of the disturbance D, which the algorithm had to compensate for.

For the first motion initial and final states were defined as

$$q_{0,1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{47}$$

$$q_{f,1} = \begin{bmatrix} 1.0\,\mathrm{m} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{48}$$

The requirement for successful termination of the ILC was

$$\|q_{d,1}(N) - q_{M,1}(N)\|_2 < 0.2 \tag{49}$$

while the parameter α in the ILC update law was set to 0.3 to provide a trade-off between rejection of white noise and speed of convergence. Applying the algorithm to the real system led to convergence in 9 iterations. Figure 4 depicts the error $q_{d,1}(t)-q_{j,1}(t)$ for different iterations j, visualizing the convergence of each state. In the first iteration the errors of x and $\dot{x}$ tended to grow during the course of the motion, while the errors of the other states oscillated about zero. In the ninth iteration the magnitude of all errors was significantly reduced. The error norm was reduced from 0.9857 after iteration 1 to 0.0922 after iteration 9.

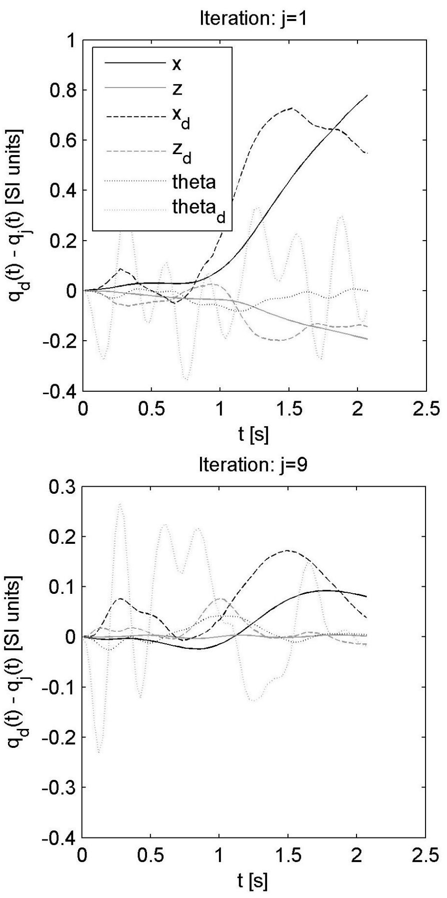

Fig. 4. Maneuver 1, State Error

The next maneuver was a side-to-side motion over 1.5 m, intended to excite more nonlinear behavior of the system.

$$q_{0,2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{50}$$

$$q_{f,2} = \begin{bmatrix} 1.5\,\mathrm{m} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{51}$$

Learning this maneuver without utilizing any previous knowledge from maneuver 1 resulted in successful termination after 8 iterations, which is comparable to maneuver 1. However, the initial transients were more severe. The vehicle was close to becoming unstable or colliding with the boundaries of the airspace. Figure 5 shows the error $q_{d,2}(t) - q_{j,2}(t)$ for different iterations j. The error norm was reduced from 0.9565 after iteration 1 to 0.1705 after iteration 8.

Maneuver 2 was then compared to the approach described in Section IV. The model was adjusted based on the results from maneuver 1 and the 1.5 m side-to-side motion was relearned, resulting in convergence after 7 iterations with a final error norm of 0.1579. While the improvement in the number of iterations can be attributed to noise in the learning process it should be noted that the initial transients were much smaller than for either maneuver 1 or 2, see Figure 6. This shows that it can be beneficial to take advantage of

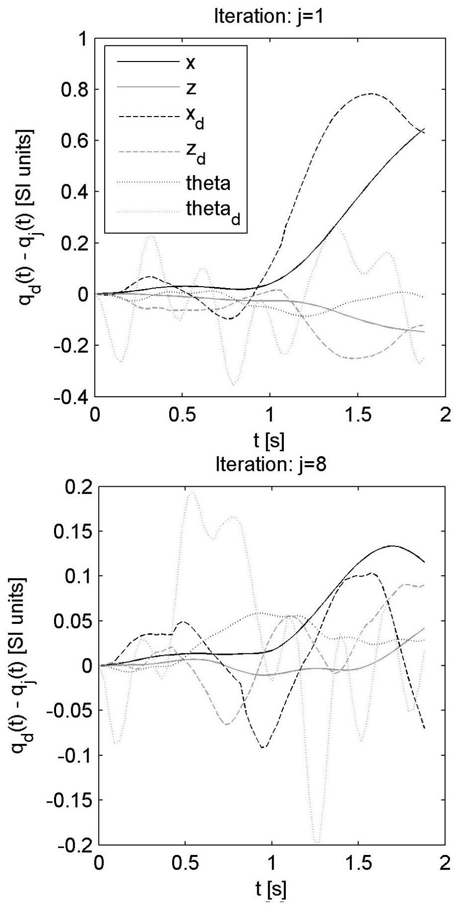

Fig. 5. Maneuver 2, State Error

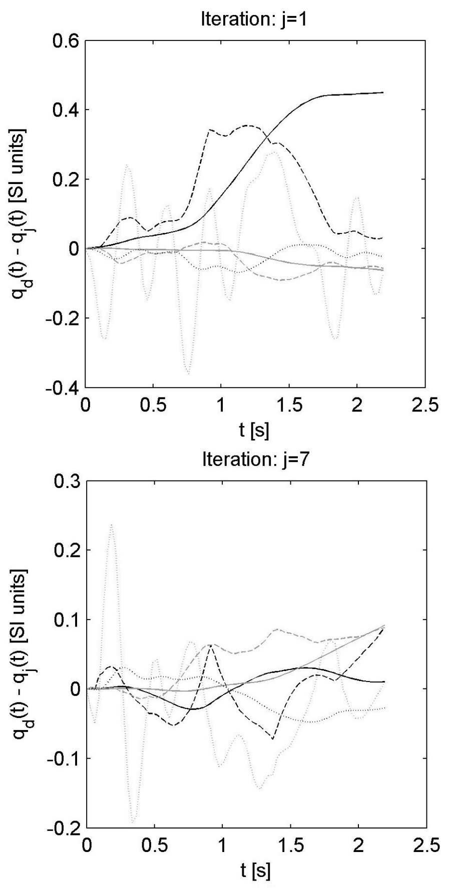

Fig. 6. Maneuver 3, State Error

previously gained information by adjusting the underlying model. The legend for Figure 6 is the same as for Figure 5.

VI. Conclusion

An algorithm has been presented which enables a system to iteratively perform an aggressive motion, given a simple model which captures the essential dynamics of the system. Expressing the problem in the lifted domain allows the synthesis of a non-causal controller, which can anticipate recurring disturbances and compensate for them by adjusting a feedforward signal. The controller synthesis is formulated as a LLS problem, which can be readily solved and executed online with modest computational resources. Using the data from a well tracked trajectory it is possible to adjust the model in order to learn a motion which is similar to the original reference. The algorithm has been successfully applied to a quadrotor UAV, reducing the error norm of the desired maneuver by an order of magnitude.

References

[1] Bristow D.A., Tharayil M., Alleyne A.G.: “A survey of iterative learning control”, IEEE Control Systems Magazine, Vol. 26, No. 3, pp. 96-114, 2006

[2] Chen Y.Q., Moore K.L.: “A Practical Iterative Learning Path-Following Control of an Omni-Directional Vehicle”, Special Issue on Iterative Learning Control, Asian Journal of Control, Vol. 4, No. 1, pp. 90-98, 2002

[3] Ghosh J., Paden B.: “Pseudo-inverse based iterative learning control for nonlinear plants with disturbances”, Proceedings of the 38th IEEE Conference on Decision and Control, Vol. 5, pp. 5206-5212, 1999, DOI 10.1109/CDC.1999.833379

[4] Chin I., Qin S.J., Lee K.S., Cho M.: “A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection”, Automatica, Vol. 40, No. 11, pp. 1913-1922, 2004

[5] Lee K.S., Lee J.H., Chin I.S., Lee H.J.: “A model predictive control technique for batch processes and its application to temperature tracking control of an experimental batchreactor”, A.I.Ch.E. Journal, Vol. 45, No. 10, pp. 21752187, 1999

[6] Lee J.H., Lee K.S., Kim W.C.: “Model-based iterative learning control with a quadratic criterion for time-varying linear systems”, Automatica, Vol. 36, pp. 641657, 2000

[7] Cho M., Lee Y., Joo S., Lee K.S.: “Semi-Empirical Model-Based Multivariable Iterative Learning Control of an RTP System”, IEEE Transactions on Semiconductor Manufacturing, Vol. 18, No. 3, pp. 430-439, 2005

[8] Rice J.K., Verhaegen M.: “Lifted repetitive learning control for stochastic LTV systems: A structured matrix approach”, Submitted to: Automatica, March, 2007

[9] de Roover D., Bosgra O.H.: “Synthesis of robust multivariable iterative learning controllers with application to a wafer stage motion system”, Int. J. Contr., vol. 73, no. 10, pp. 968979, 2000

[10] Mezghani M., Roux G., Cabassud M., Le Lann M.V., Dahhou B., Casamatta G.: “Application of iterative learning control to an exothermic semibatch chemical reactor”, IEEE Trans. Contr. Syst. Technol., vol. 10, no. 6, pp. 822834, 2002

[11] Norrlof M.: “An adaptive iterative learning control algorithm with experiments on an industrial robot”, IEEE Trans. Robot. Automat., vol. 18, no. 2, pp. 245251, 2002

[12] Schwartz A., Polak E., Chen Y.: “RIOTS 95”, Optimal Control Toolbox for Matlab V6.5