

Improved Path Following for an Omni-Directional Vehicle Via Practical Iterative Learning Control Using Local Symmetrical Double-Integration¹

YangQuan Chen** and Kevin L. Moore'

** Singapore Science Park Design Center; Seagate Technology International,
63 The Fleming, Singapore Science Park, Singapore 118249.

* Center for Self-Organizing & Intelligent Systems (CSOIS)
Dept. of Electrical and Computer Engineering
College of Engineering, Utah State University

Abstract

A new practical iterative learning control (ILC) updating law is proposed to improve the path following accuracy for an omni-directional autonomous mobile robot. The ILC scheme is applied as a feedforward controller to the existing feedback controller. By using the local symmetrical double-integral of the feedback control signal of the previous iteration, the ILC updating law takes a simple form with only two design parameters: the learning gain and the range of local integration. Convergence analysis is presented together with a design procedure. Simulation results on a difficult maneuver are presented to illustrate the effectiveness of the proposed simple and yet practical scheme. The simulation is based on the model of a novel robotic platform, the Utah State University (USU) Omni-Directional Vehicle (ODV), which uses multiple smart wheels, whose speed and direction can be independently controlled through dedicated processors for each wheel. Key words: Iterative learning control (ILC); local symmetrical double-integration (LSI^2); convergence analysis; controller design and tuning; path following; omni-directional vehicle (ODV); mobile robot.

1 Introduction

Iterative Learning Control (ILC) [1,2] is now regarded as a value-added block to enhance the feedback control performance of some classes of systems by capitalizing on the repetitiveness of these systems operation. Detailed literature reviews on ILC research can be found in [3,4]. Most of the existing work has focused on the analysis issue of ILC schemes. However, the convergence conditions found in the literature are typically not sufficient for actual ILC applications. Therefore, in recent years increasing efforts have been made on the design issue of ILC. These can be observed from the latest books [5,4] and the dedicated ILC web server [6]. A recent survey on ILC design issue [7] has documented various practically

tested design schemes, though mainly for robotic manipulators. However, to draw attention from industry, the existing design techniques are still not sufficiently attractive as compared to the successful use of PID (proportional-integral-derivative) controllers in industries. The ILC design issue should better be attacked in a similar setting way of PI/PID controller.

In this paper a new practical iterative learning control (ILC) updating law is proposed to improve the path following accuracy for an omni-directional autonomous mobile robot. By using the local symmetrical double-integral (LSI^2) of feedback control signal of the previous iteration, the ILC updating law takes a simple form with only two design parameters: the learning gain and the range of local integration. Convergence analysis is presented together with a design procedure. Some practical considerations in the parameter tuning are also outlined. Simulation studies show that the approach can be used to improve the path following accuracy during a difficult, path-tracking maneuver. The simulation is based on the model of a novel robotic platform, the Utah State University (USU) Omni-Directional Vehicle (ODV) [11,10]. It is shown that conventional controllers are unable to follow the desired path as well as desired. However, with the proposed ILC scheme much better path-following can be achieved without resorting to detailed modeling of the mobile robot.

The remainder of the paper is organized as follows. Sec. 2 formulates the control problem with a new learning updating scheme using a local symmetrical double-integral of the feedback signal. In Sec. 3, convergence analysis for linear systems is given, together with some details on the ILC design issues. Sec. 4 describes the simulation situations and representative results are presented for ODV maneuvers before and after using the proposed ILC scheme. Finally, Sec. 5 concludes the paper.

2 LSI^2 -type ILC

A block-diagram of the system and the proposed control scheme is shown in Fig. 1. In this figure FBC stands for feedback controller and y_d is the given desired output trajectory to be tracked. After the i -th iteration (repetitive operation), the feedforward control signal $u_{f,i}^1$ and the

¹The research work of Professor Kevin L. Moore (E-mail: moorek@ece.usu.edu) was conducted in part under the U.S. Army Tank-Automotive and Armaments Command (TACOM) Intelligent Mobility Program (agreement No. DAAE07-98-3-0023). Corresponding author: Dr YangQuan Chen. E-mail: yqchen@ieee.org

feedback control signal u_{fb}^i are to be stored in the memory bank for constructing the feedforward control signal at the next iteration, i.e., u_{ff}^{i+1} . The stored feedback control signal u_{fb}^i are to be locally symmetrically double-integrated (LSI²) and multiplied by a learning gain γ . Note that in Fig. 1, transfer functions of the plant and the controller are used. This is not strange because in engineering practice, to design a control system, it is very common and fundamental to have an approximate linear model $G(s)$ for frequencies below a frequency of interest, say, ω_c . In fact, for a feedback-controlled system, it is almost sure that, its frequency response can be well-approximated by a linear system, i.e., $G(s)$, the closed-loop transfer function. Therefore, at this point, it should be understood that, $G(s)$ in Fig. 1 is the major (linear) part of the *plant*, which may actually be nonlinear.

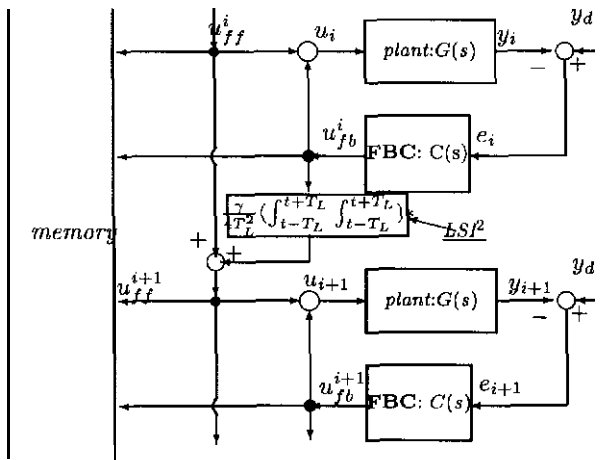


Figure 1: Block Diagram of Local Symmetrical Double-Integral-type (LSI²) Iterative Learning Control

As shown in Fig. 1, the LSI²-type learning updating law is proposed as

$$u_{ff}^{i+1}(t) = u_{ff}^i(t) + \frac{\gamma}{4T_{L1}T_{L2}} \int_{t-T_{L1}}^{t+T_{L1}} \int_{\tau-T_{L2}}^{\tau+T_{L2}} u_{fb}^i(\tau_1) d\tau_1 d\tau \quad (1)$$

while the overall control signal is simply

$$u_{i+1}(k) = u_{ff}^{i+1}(k) + u_{fb}^{i+1}(k). \quad (2)$$

In this paper, we shall consider the case for $T_{L1} = T_{L2} = T_L$. Therefore, in Fig. 1, there are two parameters: γ , the learning gain, and T_L , the width of the LSI². These two parameters are to be designed and specified. Clearly, when $T_L \rightarrow 0$, (1) reduces to the standard simple ILC update

$$u_{ff}^{i+1}(t) = u_{ff}^i(t) + \gamma u_{fb}^i(t). \quad (3)$$

It has been noted in [8] that, the above scheme will not work in practice [9]. As discussed in Sec. 3, a local integration is essential, i.e., T_L cannot be 0. The discrete-time

version of (1) can be written as

$$u_{ff}^{i+1}(k) = u_{ff}^i(k) + \frac{\gamma}{(2M+1)^2} \sum_{m=-M}^M \sum_{j=m-M}^{m+M} u_{fb}^i(k+j). \quad (4)$$

where T_s is the sampling period and $T_L/T_s = M$. In (4), a simple rectangular formula of the numerical quadrature is used which is simply a double algebraic averager a special type of noncausal filter discussed in [8, 9].

Here, our control task is to track the given desired output trajectory $y_d(t)$ over a fixed time interval $[0, T]$ as closely as possible. With an existing feedback controller $C(s)$, the main objective of this paper is to use a learning feed forward controller (LFFC) given by the ILC updating law (1) to achieve a better trajectory tracking performance. In what follows, we will perform an analysis on the ILC convergence properties of this scheme and will give a practical procedure for the design of the ILC parameters. Simulation results on the USU ODV will be presented to illustrate the effectiveness of the LSI²-ILC scheme.

3 Analysis and Design for LSI²-ILC

The convergence of the proposed learning controller is in the sense that u_{ff}^i approaches to a fixed signal as i increases and meanwhile, $y_i(t) \rightarrow y_d(t)$ over the fixed time interval $[0, T]$. This is summarized in the following theorem, where " \mathcal{L} " denotes the standard Laplace transform.

Theorem 3.1 A linear system shown in Fig. 1 is controlled by a suitable feedback controller which performs a given task repeatedly. A local symmetrical double-integral-type (LSI²) ILC scheme (1) is applied as a learning feed-forward controller (LFFC). There exists a real constant γ and a positive T_L ($0 < T_L < T$) such that the learning process is convergent and furthermore,

$$\lim_{i \rightarrow \infty} U_{ff}^i(s) \rightarrow Y_d(s)/G(s), \quad (5)$$

where $U_{ff}^i(s) = \mathcal{L}[u_{ff}^i(t)]$ and $Y_d(s) = \mathcal{L}[y_d(t)]$. The convergence rate is given by

$$\rho(\omega, \gamma, T_L) \triangleq |1 - \gamma H(\omega, T_L) G_c(j\omega)| < 1, \quad (6)$$

where $G_c(s)$ is the closed loop transfer function with $G_c(s) = C(s)G(s)/(1 + C(s)G(s))$ and $H(\omega, T_L) = \sin^2(\omega T_L)/(\omega T_L)^2$.

Theorem 3.1 implies that the iterative learning controller is essentially applied to invert the plant to be controlled in an iterative manner. Since a linear system is considered in this paper, in the sequel, frequency domain notion is used. Using the Laplace transformation, the updating law (1) becomes

$$U_{ff}^{i+1}(s) = U_{ff}^i(s) + \gamma H(\omega, T_L) U_{fb}^i(s) \quad (7)$$

where $U_{fb}^i(s) = \mathcal{L}[u_{fb}^i(t)]$, $s = j\omega$ and $H(\omega, T_L)$ is from the property of Laplace transformation, i.e.,

$$\mathcal{L}\left\{ \frac{1}{2T_L} \int_{t-T_L}^{t+T_L} \left[\frac{1}{2T_L} \int_{\tau-T_L}^{\tau+T_L} u_{fb}(\tau_1) d\tau_1 \right] d\tau \right\}$$

$$\triangleq H(\omega, T_L)U_{fb}(s). \quad (8)$$

Now we proceed to present a proof of Theorem 3.1. Proof: From Fig. 1, the feedback signal can be written as

$$U_{fb}(s) = -G_c(s)U_{ff}(s) + G_c(s)Y_d(s)/G(s). \quad (9)$$

The learning updating law (7) becomes,

$$U_{ff}^{i+1}(s) = [1 - \gamma H(\omega, T_L)G_c(s)]U_{ff}^i(s) + \gamma H(\omega, T_L)G_c(s)Y_d(s)/G(s). \quad (10)$$

Iterating (10), one obtains

$$U_{ff}^{i+1}(s) = [1 - \gamma H(\omega, T_L)G_c(s)]^i U_{ff}^0(s) + \{1 - [1 - \gamma H(\omega, T_L)G_c(s)]^i\} Y_d(s)/G(s). \quad (11)$$

Since $H(\omega, T_L)$ and $G_c(s)$ essentially have low pass filter characteristics, it is clearly possible to choose a suitable γ such that (6) is true. In addition, T_L can be used to shape $\rho(\omega, \gamma, T_L)$ which is the convergence rate in (11). Therefore, there exists a design of γ and T_L such that (6) holds and from (11) $\lim_{i \rightarrow \infty} U_{ff}^i(s) \rightarrow Y_d(s)/G(s)$ and moreover, $y_i(t) \rightarrow y_d(t)$ for all $t \in [0, T]$ as $i \rightarrow \infty$.

Remark 3.1 Let $H'(\omega, T_L) = \frac{\sin(\omega T_L)}{\omega T_L}$. It is clear that both $H'(\omega, T_L)$ and $H(\omega, T_L)$ are derived from local symmetrical integration. In terms of frequency response, we can see from Fig. 2 that $H(\omega, T_L)$ is between $H'(\omega, 2T_L)$ and $H'(\omega, T_L)$ of the lower frequency hand. However, the high frequency response of $H(\omega, T_L)$ are much better than that of $H'(\omega, T_L)$. Therefore, $H(\omega, T_L)$ is more preferred than $H'(\omega, T_L)$ in practice. That is, the learning updating law (1) is more efficient in handling high frequency uncertainty or disturbance.

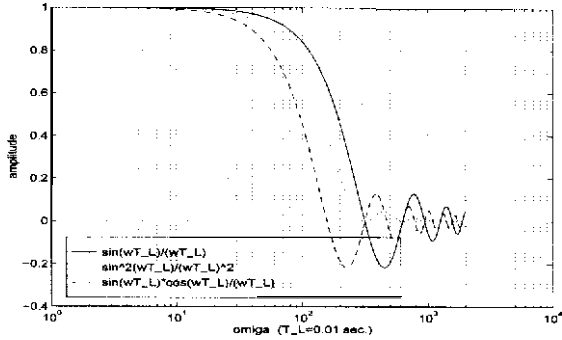


Figure 2: Frequency response comparison of $H'(\omega, T_L)$ and $H(\omega, T_L)$ ($T_L = 0.01$ sec.)

The analysis of LSI²-ILC is based on the prior knowledge we assumed: 1) the frequency of the desired trajectory, which is less than a known frequency denoted by ω_d and $\omega_d < \omega_c$, where ω_c is the systems cut-off frequency, and 2) an estimate of $G_c(j\omega_c)$ or $G_c(j\omega_d)$. Clearly, the above knowledge is minimal for controller design.

Design Method for T_L . T_L is used to control

the high frequency signal components stored in the memory bank. These high frequency signal components, if not well handled, may be accumulated due to different phase relationship from iteration to iteration. This is the major reason of the divergence for some ILC schemes, which may be convergent at the initial iterations, but in which, as the ILC scheme runs longer and longer, divergence can be observed in practical applications [7]. However, too large a value of T_L deteriorates the signal and low frequency components when smoothing out the high frequency components. Therefore, a suitably chosen T_L is very important. A simple consideration is that, the signal energy can not be attenuated by half via $H(\omega, T_L)$. One can set

$$H(\omega_d, T_L) = \frac{\sqrt{2}}{2} \quad (12)$$

which will give an estimate of T_L by

$$\omega_d T_L \approx 1.002 \quad (13)$$

using `solve('sin(x)/x=sqrt(sqrt(2)/2)')` of MATLAB Symbolic Math Toolbox. Therefore,

$$T_L \geq \frac{1.002}{\omega_d}. \quad (14)$$

An alternative practical design procedure can be like this. Because at the first iteration only the feedback controller is commissioned, at the end of the first iteration, perform a discrete Fourier transform (DFT) of the feedback signal $u_{fb}^0(t)$. This gives the spectrum information of the feedback signal. A frequency ω_c' can then be chosen to be a little bit higher than the frequency corresponding to the magnitude peak in the amplitude-frequency plot of $u_{fb}^0(t)$. Then, one can use this ω_c' to obtain a design of T_L from (14).

Since in real world, ω_c or ω_d is finite, clearly, according to (14), T_L cannot be zero in practice.

Design Method for γ . Firstly, during practice, one can always start with a smaller, conservative γ via which the learning process converges. Then fine tuning of γ is still possible based on partial knowledge of $G_c(j\omega)$.

It is assumed that at least the value of $G_c(j\omega_c)$ is available. Let $G(j\omega) = A(\omega)e^{j\theta(\omega)}$. Denote $\bar{\rho} = \rho^2(\omega, \gamma, T_L)$. Then, from (6),

$$\bar{\rho} = 1 - 2\gamma H(\omega, T_L)A(\omega) \cos \theta(\omega) + \gamma^2 H^2(\omega, T_L)A^2(\omega). \quad (15)$$

Clearly, γ should be chosen to minimize $\bar{\rho}$. From (15), the best γ should be

$$\gamma = \frac{\cos \theta(\omega)}{H(\omega, T_L)A(\omega)} \quad (16)$$

by setting $\frac{d\bar{\rho}}{d\gamma} = 0$. At frequency ω_c ,

$$\gamma = \frac{\cos \theta(\omega_c)}{H(\omega_c, T_L)A(\omega_c)}. \quad (17)$$

It should be noted that γ may be negative at certain frequency range. For most applications, ω_d is quite small and in this case γ can be given approximately by

$$\gamma \leq \sqrt{2}. \quad (18)$$

When only $G_c(j\omega_d)$ is known, γ can be designed similarly according to (16). If the knowledge of $G_c(j\omega)$ within a frequency range $[\omega_L, \omega_H]$ is known, then a plot of $\gamma(\omega)$ is available from (17). This plot is useful in selecting a suitable γ when different frequencies of interest are to be considered in $[\omega_L, \omega_H]$.

In any case, it is possible to tune γ to make the ILC convergence as fast as possible. However, there exists a limit on the ILC convergence rate. It is quite clear that by substituting (16) into (15), the corresponding minimal $\bar{\rho}$ is given by

$$(P)\gamma_{\min} = \sin S(w), \quad (19)$$

i.e.,

$$\rho \geq |\sin \theta(\omega)| = \rho^* \quad (20)$$

which implies that, for a given ω of interest, the convergence rate cannot be faster than a limit characterized by ρ^* . This limit is independent of the learning scheme applied. The only way to achieve a faster ILC convergence process is to design the feedback controller $C(j\omega)$ such that the phase response of the closed-loop system will be well-behaved as required in (19).

To achieve a faster ILC convergence, we could use some heuristic or rule-based ideas. For example, we can reevaluate T_L at the end of every iteration. This will not cost a lot but can keep a tight monitoring of possible variations of the system dynamics and the uncertainty/disturbance. Alternatively, we can start ILC with a smaller γ and increase γ when the tracking error keeps decreasing and decrease γ while the tracking error keeps increasing. Another way is to use a cautious (larger) T_L at the beginning of ILC process and then decrease T_L when the ILC scheme converges to a stage with little improvement. In this case, a smaller T_L leaves more high frequency components of the feedback control signal in the memory bank. This in turn may further improve the convergence performance.

In the next section we shall apply the proposed LST²-ILC scheme to improve the path following accuracy for an omni-directional autonomous mobile robot (the USU ODV) with minimal modeling efforts.

4 Simulation Results on USU ODV

T1 USU ODV. The simulation is based on a MATLAB Simulink model of the T1 mobile robotic platform with six wheels, a member of the Utah State University (USU) Omni-Directional Vehicle (ODV) family [10, 11, 12]. The USU ODV concept features multiple "smart wheels." Both the speed and direction of each smart wheel can be independently controlled through dedicated processors for each wheel, resulting in the ability to achieve complete control of the vehicle's orientation and motion in a plane—a "hovercraft on wheels."

Controller Configuration for Enhanced Mobility. The T1 uses a three-level, hierarchical control scheme that results in a multi-resolution behavior generation strategy [12]. At the coarsest resolution level, the strategy is characterized by a task decomposition approach in which a knowledge-based planner and an A*-optimization algorithm are used to specify the vehicle's path as a sequence of basic maneuvers. At the medium-resolution level, or the vehicle command

level, the basic maneuvers are converted to time-domain trajectories and a nonlinear controller is used to generate low-level setpoints to force the vehicle to track the required path in an inertial reference frame. At the finest resolution level, classical control is used to drive the vehicle's wheels to the required speed and angle setpoints.

Mission Planner. The mission planner is run once when the initial goals and the map are entered, then again whenever new goals are entered during mission execution "when sensors provide significant changes to the initial map. The system differs from traditional path planning systems in that the omni-directional vehicles controlled require not only paths to follow, but also maneuvers to execute while following the paths. The approach taken here is to identify possible maneuvers as a set of parameterized commands (called Z-commands) and then specify paths as a sequence of instantiated commands. For example, since the omni-directional vehicle is capable of controlling its body orientation independent of its direction of travel, one command (called translate) specifies travel along a straight path while maintaining a fixed vehicle body orientation. Another command (called translate-with-spin) specifies travel along a straight path while rotating the body orientation through a given angle. Other commands include curve-with-spin, spin-in-place, and curve-with-fixed-orientation. The task of the planner is to determine optimal command sequences for the vehicles such that the goals are achieved in the shortest time.

Trajectory Generator. The trajectory generator is used to convert a sequence of basic maneuvers into time signals for the desired motion. The mission and path planner generates an appropriate sequence of tasks called *basic maneuvers* required to achieve a goal. A protocol for describing and transmitting such commands, as well as a number of other important tasks for communication between the mission and path planner and other vehicle functions has been established. This protocol is the basis for optimization at the path planning level. After the path planner has determined a proper script of basic maneuvers, or Z-commands, the trajectory generator parses and filters this script, to produce time-domain signals. Also, from a control-theoretic perspective, ideally the trajectory generator should decompose the sequences of basic maneuvers into time functions that are a linear combination of step, ramp, decaying exponential, and sinusoidal functions.

Outer-Loop Control and Wheel-level Control. The closed-loop, or outer-loop, control algorithm block compares the desired trajectory to the actual trajectory (in inertial space) and computes appropriate signals $(\dot{x}, \dot{y}, \dot{\theta})$ in inertial coordinates to be used by the wheel movement routine. This computation is done on the basis of errors in the vehicle's inertial position. Vehicle motion is determined either through odometry or GPS measurements. Finally, the wheel movement routine block uses the kinematic relationships of the vehicle to translate $(\dot{x}, \dot{y}, \dot{\theta})$ into separate drive (speed) and steering angle (direction) commands, v_i and θ_i , respectively for use by the low-level controllers. After the path-tracking controller generates appropriate speed and steering commands, PID (proportional-integral-derivative) wheel-level

controllers are used to actually force the wheels to the correct speed and steering angle. Encoder feedback is used to compute error signals for the PID controllers. The vehicle-level control algorithm is concerned with the following problem: given a desired vehicle motion (x, y , and yaw θ), determine the appropriate velocities (speed and direction) for each wheel. In [11], the modeling and controller derivation to solve the path-tracking controllers problem was described in detail. The basic idea is to define an inertial to body reference transformation by the pseudo-Euler transformation and then the system can be modeled as a typical robotic system. Using an exact feedback linearization control scheme, given any desired force vector u we can uniquely define the required velocity and direction for drive and steering motors of each wheel. While for any desired path planning, there exists a desired force vector u which can be obtained by resolving the forces produced by the drive motors into the direction they have been pointed by the steering motors. Thus, the drive motors are effectively a gain, which is easily handled by the controller (and is included in the simulation). The actual approach used to control the vehicle is a cascade architecture, as shown in Fig. 3. Because the open-loop system seen by the outer-loop controller system looks like a single integrator with a unity-gain low-pass filter, the controllers were chosen to be proportional-integral (PI) algorithms in order to achieve zero steady-state error for ramp inputs. Also in Fig. 3 a computed-torque feedforward velocity term is used to speed up the convergence to the desired path. A close-up of outer-loop controller (marked control. c in Fig. 3) is shown in Fig. 4) which includes a PI-controller, a feedforward control summing point and a set-point (SP) generator for steering and driving motors control of the six smart wheels. A complete Matlab/Simulink simulation has been built which is based on a cascade architecture shown [11]. The model has been validated through experiments [11]. Better control in the outer-loop shown in Fig. 3 is the major concern of this paper.

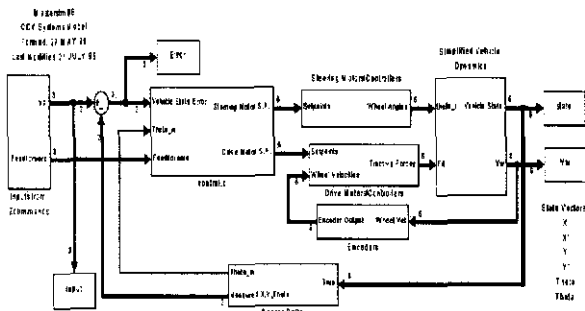


Figure 3: Cascade feedback control of USU ODV.

Simulation Example and PI Control. A difficult path tracking problem is defined by the following sequence of Z-commands:

```

translatewithspin(10,0,1,18);
translatewithspin(0,10,1.414,54);
translatewithspin(10,10,1,90);

```

where $\text{TranslateWithSpin}(\text{final}_x, \text{final}_y, \text{velocity}, \text{final_orientation})$ is one of the protocol commands designed to translate the vehicle to the point

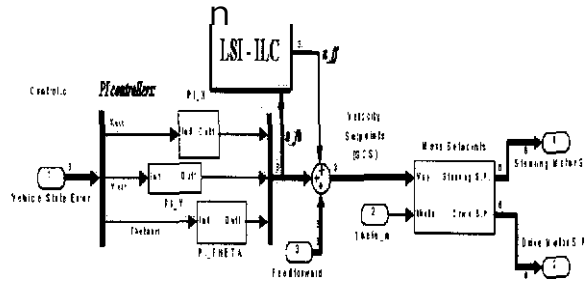


Figure 4: Iterative Learning Control in the outer-loop with PI controller.

$(\text{final}_x, \text{final}_y)$ at the specified velocity and spin the vehicle to achieve the specific final_orientation during the move. The path defined above is shown in Fig. 5. PI-controllers were used in the outer-loop path-tracking controller for $(x, y, 0)$ as shown in Fig. 4, based on the linearized model of T1. 4 typical set of PI control results (with the computed torque FF term) are shown in Fig. 6. We can see that the tracking errors are large at the corner points of the path.

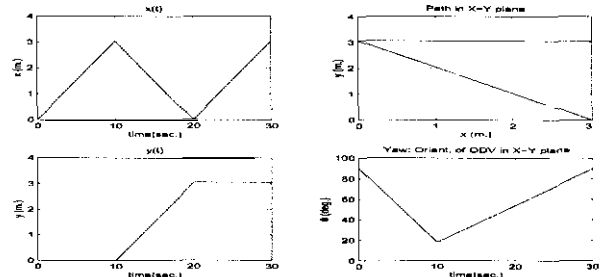


Figure 5: Z-path.

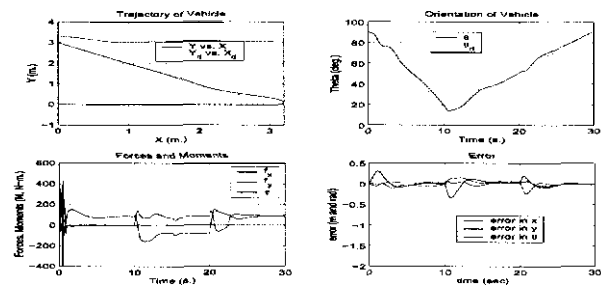


Figure 6: PI control results for Z-path following (with FF).

Results Using LSI²-ILC. The LSI²-ILC scheme outlined in Fig. 1 was applied to the system through the LSI²-ILC feedforward block in the outer-loop of the T1 ODV control system, as shown in Fig. 4. In our ILC-based control system we did not use the computed-torque feedforward term that is used in the non-ILC controller, because the point of ILC is to derive the proper feedforward control input to add to the system. In the simulation the discrete LSI² is realized simply by applying `filtfilt`, the zero-phase filter in MATLAB, twice for the output signals from blocks PI-X, PI-Y,

Table 1: Standard deviation of tracking errors

Iter.	$1\sigma(e_x)$	$1\sigma(e_y)$	$1\sigma(e_\theta)$
0	0.1852	0.1057	0.1436
1	0.1086	0.0573	0.0810
2	0.0878	0.0508	0.0629
3	0.0768	0.0378	0.0535
4	0.0679	0.0323	0.0471
5	0.0596	0.0314	0.0438

PI-THETA. Note that here the filter is a FIR, i.e., $a = 1$ and $\mathbf{b} = [1, \cdot, 1]/N$ where N is the length of \mathbf{b} . N is related to T_L and the sampling period. In our simulation (step size 0.01 s), also from Fig. 6, we found that the bandwidth is not very high and we can choose a larger N , for example, from 40 to 80 or even larger. The learning gain γ is set by rule. In our example, we use $\gamma = 0.5$ after the first run (initial feedforward is set to 0) and later on, we use a fixed γ of 0.25. Since we can use the pre-stored data noncausally, here we added a phase advance of 40 steps to get a better convergence. We can see that after 5 iterations the path following performance is improved as shown in Fig. 7. Clearly the performance at the corners is improved. This improvement is quantified by the tabulation of the monotonic learning convergence shown in Table 5, which lists the standard deviation of tracking errors at each iteration.

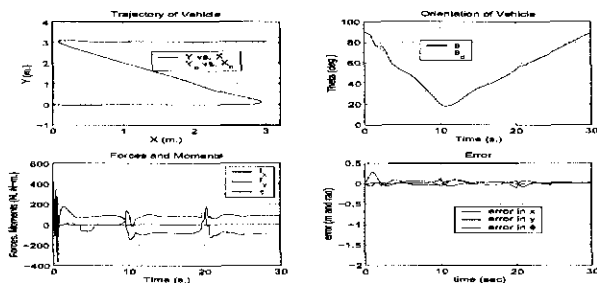


Figure 7: LSI²-ILC results for Z-path following with phase advance.

5 Concluding Remarks

We have proposed a new practical iterative learning control updating law, LSI²-ILC, using the idea of local symmetrical double-integration of the feedback control signal of the previous iteration. The updating law takes a simple form with only two design parameters: the learning gain and the range of local double-integration. Convergence analysis is presented together with a design procedure. Some practical considerations in the parameter tuning are also outlined. Simulation results based on the model of the Utah State University (USU) Omni-Directional Vehicle (ODV) are presented to demonstrate that 1) the proposed LSI²-ILC is effective for improving path-tracking performance, and 2) the scheme is simple and practical to apply.

References

[1] Arimoto, S., Kawamura, S., and Miyazaki, F. (1984). Bettering operation of robots by learning. *J. Of Ro-*

botic Systems, **1**(2), 123-140.

- [2] Moore K. L. (1993). *Iterative learning control for deterministic systems*. Advances in Industrial Control. Springer-Verlag.
- [3] K. L. Moore, "Iterative learning control - an expository overview," *Applied & Computational Controls, Signal Processing, and Circuits*, 1998. Also available at <http://www.engineering.usu.edu/ece/faculty/moorek/survey.zip>.
- [4] Chen, Y. and Wen, C. (1999). *Iterative learning control: convergence, robustness and applications*. Lecture Notes series on Control and Information Science. Vol. LNCIS-248. Springer-Verlag.
- [5] Bien, Z. and Xu, J.-X. (1998). *Iterative Learning Control Analysis, Design, Integration and Applications*. Kluwer Academic Publishers.
- [6] Chen, Y. (1998). Dedicated web server for Iterative Learning Control research. <http://ilc.ee.nus.edu.sg/>. Mirrored at <http://cicserver.ee.nus.edu.sg/~ilc>.
- [7] Longman, R. W. (1998). "Designing Iterative Learning and Repetitive Controllers" In Z. Bien and Xu J.-X. eds, "*Iterative Learning Control Analysis, Design, Integration and Application*" Kluwer Academic Publishers, pp. 107-145.
- [8] Chen, Y., Lee, T. H., Xu, J.-X., and Yamamoto, S. (1998b). Noncausal filtering based design of iterative learning control. In K. L. Moore, editor, *Proc. of the First Int. Workshop on Iterative Learning Control*, pages 63-70, Hyatt Regency, Tampa, FL, USA.
- [9] Tan K. K., Dou H. F., Chen Y. Q. and Lee T. H. (2000). "High Precision Linear Motor Control Via Artificial Relay Tuning and Zero-Phase Filtering Based Iterative Learning" to appear in the IEEE Trans. of Control Systems Technology.
- [10] Poulson, Jacob, Gunderson, and Abbot (1998). "Design of a robotic vehicle with self-contained intelligent wheels," in Proc. of SPIE Conference on Robotic and Semi-Robotic Ground Vehicle Technology, Vol. 3366, 15-16 April, pp. 68-73, Orlando, FL, USA.
- [11] Moore, K. L. and Davidson M. (2000). "Modeling and control of a six-wheeled autonomous robot." In: *Proc. of the American Control Conference: Chicago, USA*.
- [12] Moore, K. L. and Flann M. (1999). "Hierarchical task decomposition approach to path planning and control for an omni-directional autonomous mobile robot," in *1999 IEEE Int. Symp. on Intelligent Control/Intelligent Systems and Semiotics*. Cambridge, MA.
- [13] Moore, K. L. and Bahl, V. (1999). "An iterative learning control technique for mobile robot, path-tracking control," in *Proc. of SPIE Conference on Robotic and Semi-Robotic Ground Vehicle Technology*.