IEEE ICMA 2006 Tutorial Workshop:

– Iterative Learning Control –
Algebraic Analysis and Optimal Design

Presenters:

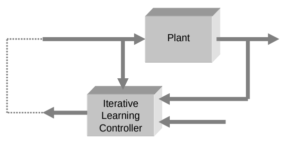
Kevin L. Moore – Colorado School of Mines
YangQuan Chen – Utah State University

Contributor:

Hyo-Sung Ahn – ETRI, Korea

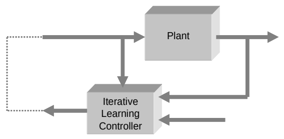
IEEE 2006 International Conference on Mechatronics and Automation
LuoYang, China

25 June 2006

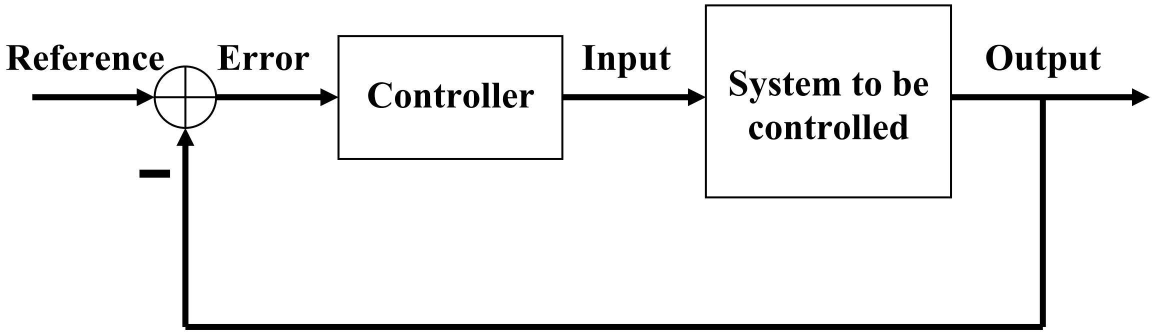


Outline

- **Introduction**
  - **Control System Design: Motivation for ILC**
  - **Iterative Learning Control: The Basic Idea**
  - **Some Comments on the History of ILC**
  - **ILC Problem Formulation**
- The "Supervector" Notation
- The w-Transform: "z-Operator" Along the Repetition Axis
- ILC as a MIMO Control System
  - Repetition-Domain Poles
  - Repetition-Domain Internal Model Principle
- The Complete Framework
  - Repetition-Varying Inputs and Disturbances
  - Plant Model Variation Along the Repetition Axis



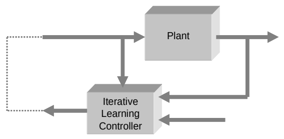
Control Design Problem



Given: System to be controlled.

Find: Controller (using feedback).

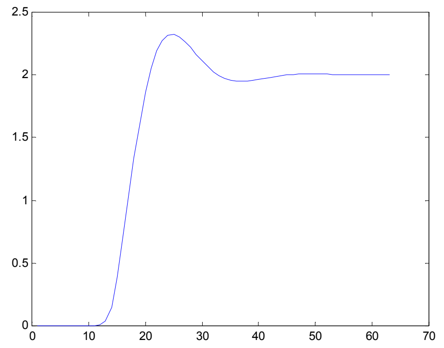
Such that: 1) Closed-loop system is stable.
2) Steady-state error is acceptable.
3) Transient response is acceptable.



Motivation for the Problem of Iterative Learning Control

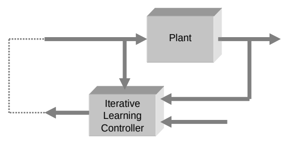
- Transient response design is hard:

  1) Robustness is always an issue:
     - Modelling uncertainty.
     - Parameter variations.
     - Disturbances.
  2) Lack of theory (design uncertainty):
     - Relation between pole/zero locations and transient response.
     - Relation between Q/R weighting matrices in optimal control and transient response.
     - Nonlinear systems.

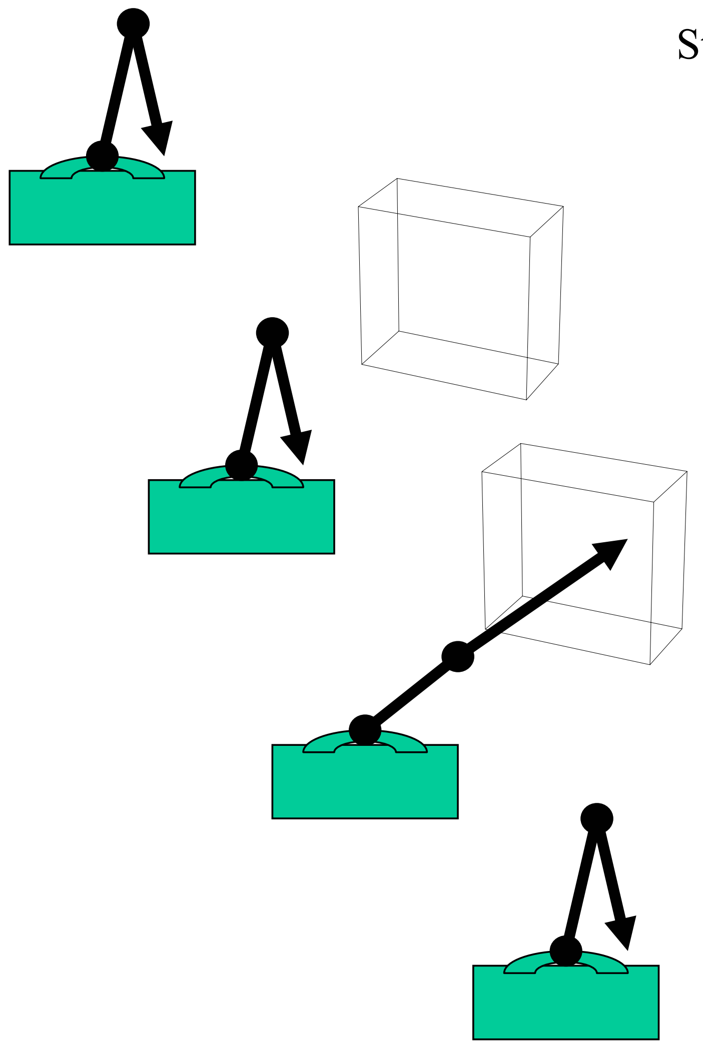


- Many systems of interest in applications are operated in a repetitive fashion.

- Iterative Learning Control (ILC) is a methodology that tries to address the problem of transient response performance for systems that operate repetitively.



Systems that Execute the Same Trajectory Repetitively

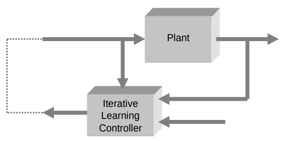


Step 1: Robot at rest, waiting for workpiece.

Step 2: Workpiece moved into position.

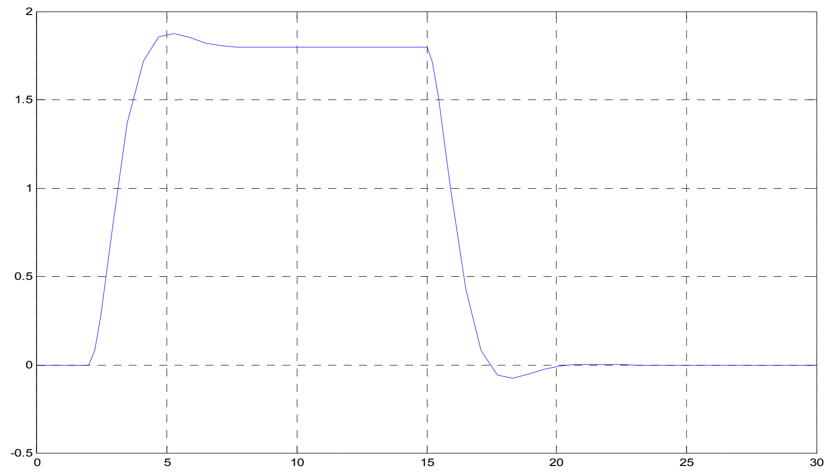
Step 3: Robot moves to desired location

Step 4: Robot returns to rest and waits for next workpiece.



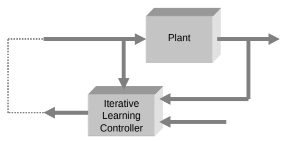
Errors are Repeated When Trajectories are Repeated

•A typical joint angle trajectory for the example might look like this:



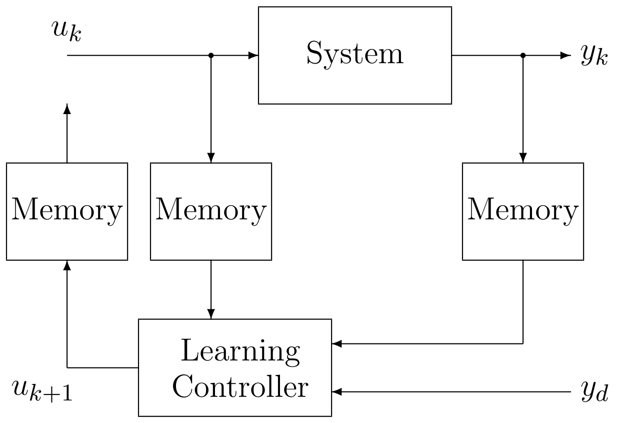
•Each time the system is operated it will see the same overshoot, rise time, settling time, and steady-state error.

•Iterative learning control attempts to *improve the transient response* by *adjusting the input to the plant during future* system operation based on the *errors observed during past* operation.

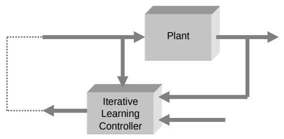


Iterative Learning Control

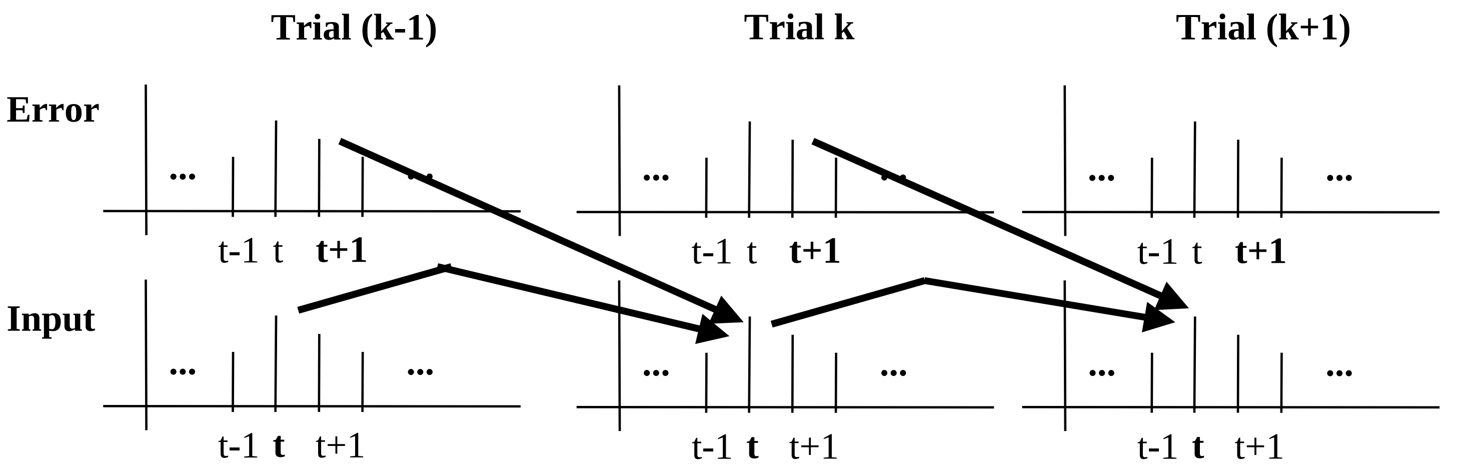
- Standard iterative learning control scheme:



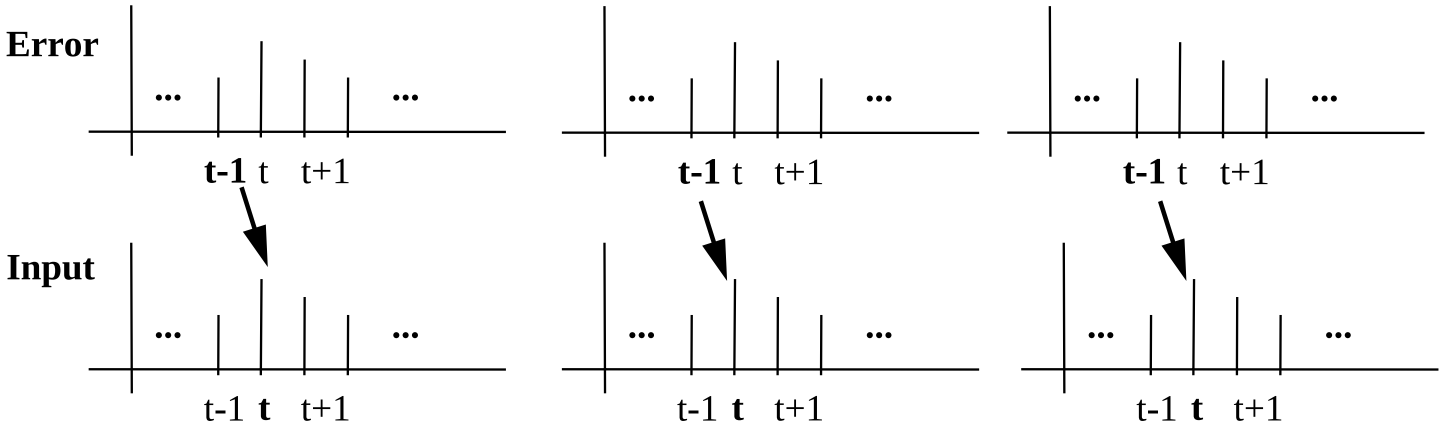
- A typical ILC algorithm has the form: $u_{k+1}(t) = u_k(t) + \gamma e_k(t+1)$.
- Standard ILC assumptions include:
  - Stable dynamics or some kind of Lipschitz condition.
  - System returns to the same initial conditions at the start of each trial.
  - Each trial has the same length.



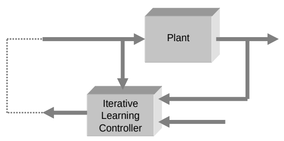
Part 1: Introduction to the Algebraic Approach to ILC



(a) ILC: $u_{k+1}(t) = u_k(t) + f(e_k(t+1))$



(b) Conventional feedback: $u_{k+1}(t) = f(e_{k+1}(t-1))$

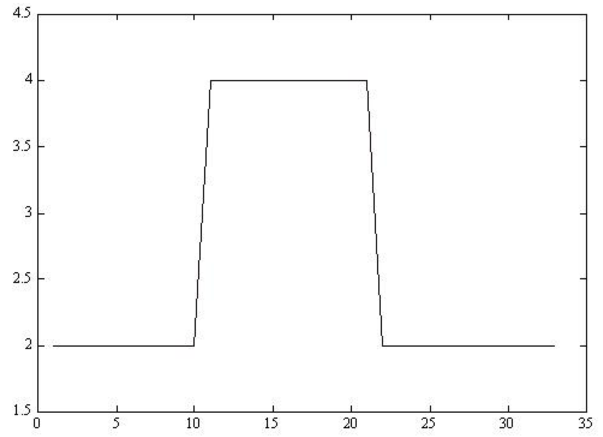


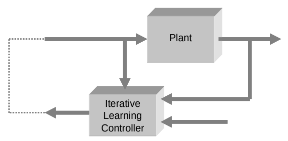
Example 1

- Consider the plant:

$$\begin{aligned} y(t+1) &= -.7y(t) - .012y(t-1) + u(t) \\ y(0) &= 2 \\ y(1) &= 2 \end{aligned}$$

- We wish to force the system to follow a signal y_d:



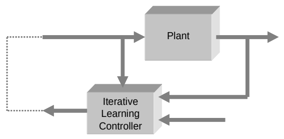


Example 1 (cont.)

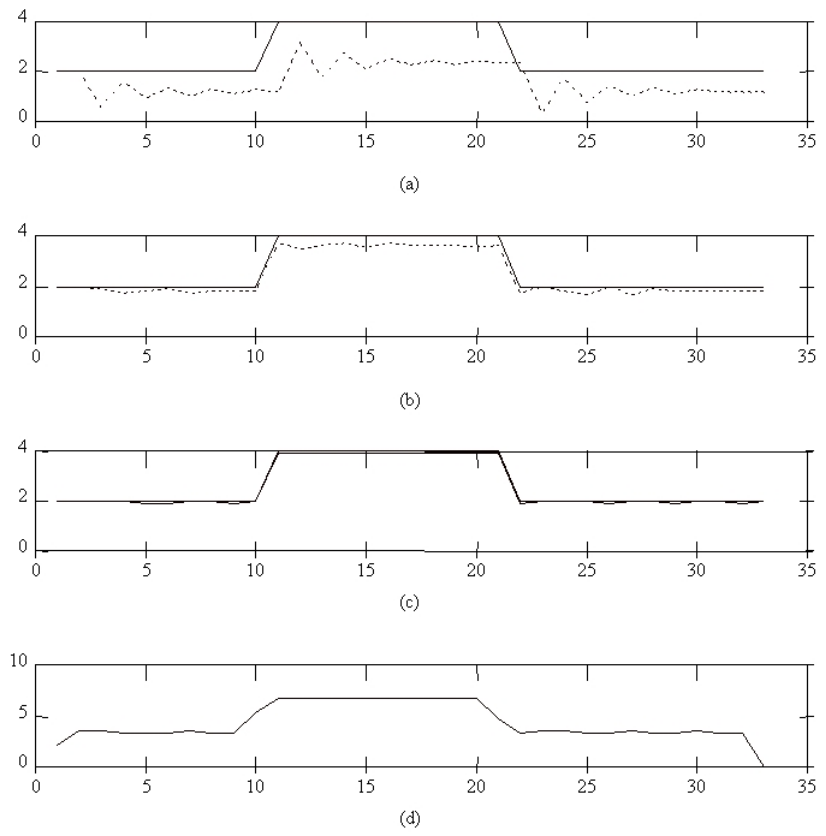
- Use the following ILC procedure:

  1. Let
  $$u_0(t) = y_d(t)$$
  2. Run the system
  3. Compute
  $$e_0(t) = y_d(t) - y_0(t)$$
  4. Let
  $$u_1(t) = u_0(t) + 0.5e_0(t+1)$$
  5. Iterate

- Each iteration shows an improvement in tracking performance (plot shows desired and actual output on first, 5th, and 10th trials and input on 10th trial).



Example 1 (cont.)



Example 2

- For the nominal plant:

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} -0.8 & -0.22 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u_k \\ y_k &= [1, 0.5] x_k \end{aligned}$$

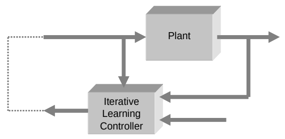
- Track the reference trajectory:

$$Y_d(j) = \sin(8.0j/100)$$

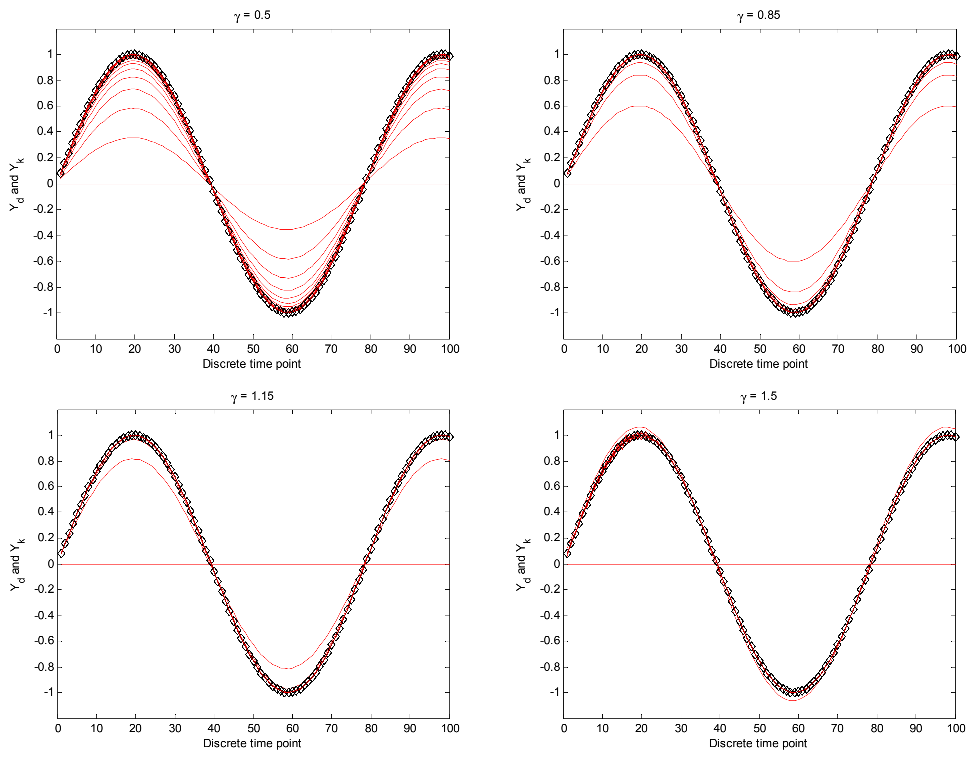
- We use the standard "Arimoto" algorithm:

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1)$$

with four different gains: $\gamma = 0.5$, $\gamma = 0.85$, $\gamma = 1.15$, $\gamma = 1.5$



Example 2 (cont.)



- For each gain, the ILC algorithm converges, but the convergence rate depends on γ.
- Without knowing an accurate model of the plant, we achieve "perfect" tracking by iteratively updating the input from trial to trail.

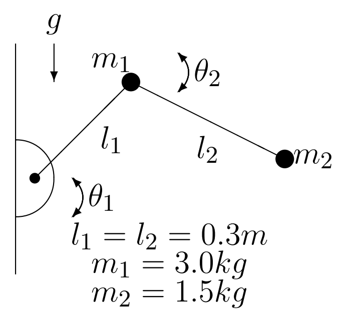
Example 3

- Consider a simple two-link manipulator modelled by:

$$A(x_k)\ddot{x}_k + B(x_k, \dot{x}_k)\dot{x}_k + C(x_k) = u_k$$

where

$$\begin{aligned}
x(t) &= (\theta_1(t), \theta_2(t))^T \\
A(x) &= \begin{pmatrix} .54 + .27\cos\theta_2 & .135 + .135\cos\theta_2 \\ .135 + .135\cos\theta_2 & .135 \end{pmatrix} \\
B(x, \dot{x}) &= \begin{pmatrix} .135\sin\theta_2 & 0 \\ -.27\sin\theta_2 & -.135(\sin\theta_2)\dot{\theta}_2 \end{pmatrix} \\
C(x) &= \begin{pmatrix} 13.1625\sin\theta_1 + 4.3875\sin(\theta_1 + \theta_2) \\ 4.3875\sin(\theta_1 + \theta_2) \end{pmatrix} \\
u_k(t) &= \text{vector of torques applied to the joints}
\end{aligned}$$



Example 3 (cont.)

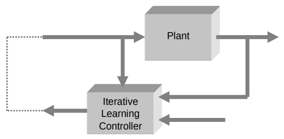
- Define the vectors:

$$\begin{aligned} y_k &= (x_k^T, \dot{x}_k^T, \ddot{x}_k^T)^T \\ y_d &= (x_d^T, \dot{x}_d^T, \ddot{x}_d^T)^T \end{aligned}$$

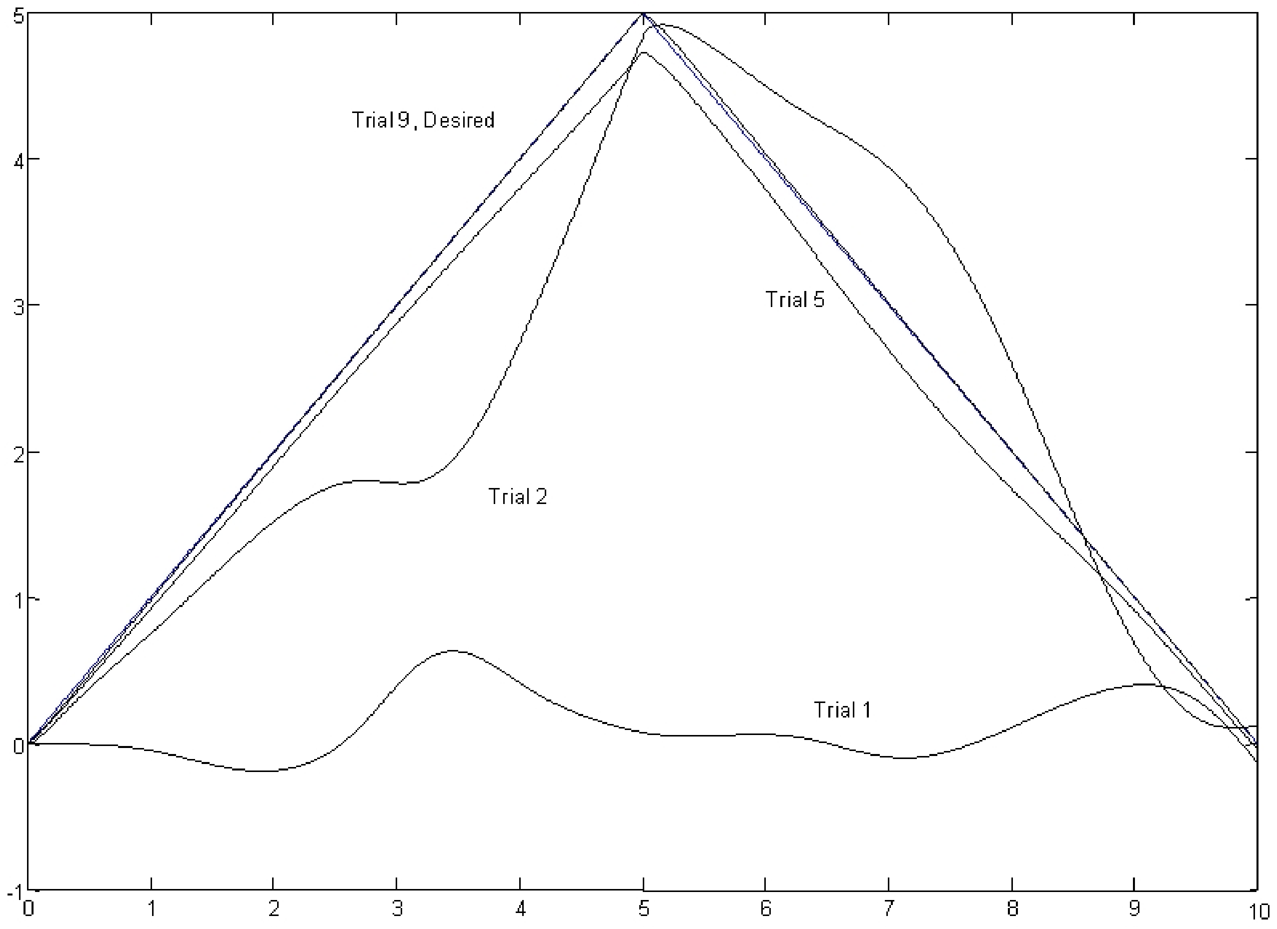
- The learning controller is defined by:

$$\begin{aligned} u_k &= r_k - \alpha_k \Gamma y_k + C(x_d(0)) \\ r_{k+1} &= r_k + \alpha_k \Gamma e_k \\ \alpha_{k+1} &= \alpha_k + \gamma \|e_k\|^m \end{aligned}$$

- Γ is a fixed feedback gain matrix that has been made time-varying through the multiplication by the gain α_k.
- r_k can be described as a time-varying reference input. $r_k(t)$ and adaptation of α_k are effectively the ILC part of the algorithm.
- With this algorithm we have combined conventional feedback with iterative learning control.

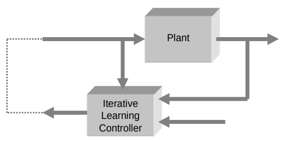


Example 3 (cont.)

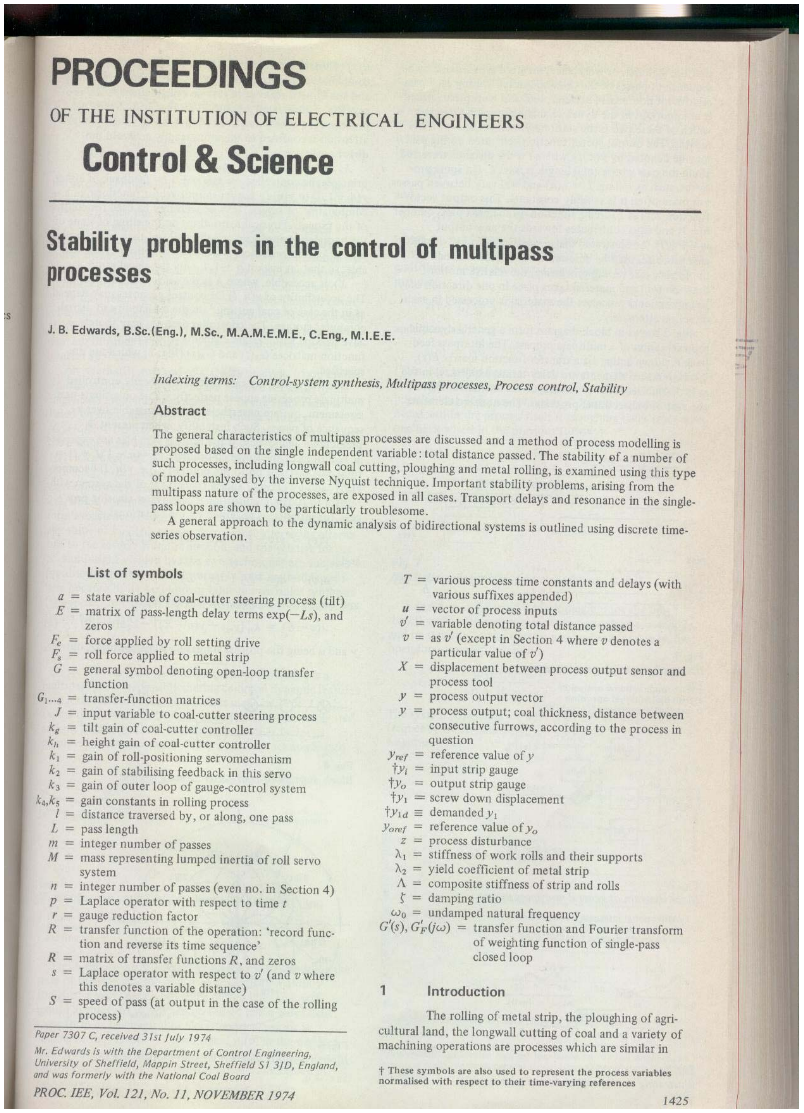


ILC History

- ILC surveys:
  - K. L. Moore, M. Dahleh, and S. P. Bhattacharyya. Iterative learning control: a survey and new results. *J. of Robotic Systems*, 9(5):563–594, 1992.
  - K. L. Moore. Iterative learning control - an expository overview. *Applied & Computational Controls, Signal Processing, and Circuits*, 1(1):151–241, 1999.
  - H. S. Ahn, Y. Q. Chen, and K. L. Moore. Iterative learning control: brief survey and categorization 1998 – 2004. *IEEE Trans. on Systems, Man, and Cybernetics*, Accepted to appear.
  - recent CSM paper by Allyene
- Pioneering work:
  - United States Patent 3,555,252 – Learning Control of Actuators in Control Systems," filed 1967, awarded 1971, learned characteristics of actuators and used this knowledge to correct command signals.
  - J. B. Edwards. Stability problems in the control of linear multipass processes. *Proc. IEE*, 121(11):1425–1431, 1974.
  - M. Uchiyama. Formulation of high-speed motion pattern of a mechanical arm by trial. *Trans. SICE (Soc. Instrum. Contr. Eng.)*, 14(6):706–712(in Japanese), 1978.
  - S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of robots by learning. *J. of Robotic Systems*, 1(2):123–140, 1984.



Edwards, Proc. IEE (1974)



PROCEEDINGS
OF THE INSTITUTION OF ELECTRICAL ENGINEERS
Control & Science

Stability problems in the control of multipass processes

J. B. Edwards, B.Sc.(Eng.), M.Sc., M.A.M.E.M.E., C.Eng., M.I.E.E.

Indexing terms: *Control-system synthesis, Multipass processes, Process control, Stability*

Abstract

The general characteristics of multipass processes are discussed and a method of process modelling is proposed based on the single independent variable: total distance passed. The stability of a number of such processes, including longwall coal cutting, ploughing and metal rolling, is examined using this type of model analysed by the inverse Nyquist technique. Important stability problems, arising from the multipass nature of the processes, are exposed in all cases. Transport delays and resonance in the single-pass loops are shown to be particularly troublesome.

A general approach to the dynamic analysis of bidirectional systems is outlined using discrete time-series observation.

List of symbols

a =	state variable of coal-cutter steering process (tilt)
E =	matrix of pass-length delay terms $\exp(-Ls)$, and zeros
F_e =	force applied by roll setting drive
F_s =	roll force applied to metal strip
G =	general symbol denoting open-loop transfer function
$G_{1...4}$ =	transfer-function matrices
J =	input variable to coal-cutter steering process
k_g =	tilt gain of coal-cutter controller
k_h =	height gain of coal-cutter controller
k_1 =	gain of roll-positioning servomechanism
k_2 =	gain of stabilising feedback in this servo
k_3 =	gain of outer loop of gauge-control system
k_4, k_5 =	gain constants in rolling process
l =	distance traversed by, or along, one pass
L =	pass length
m =	integer number of passes
M =	mass representing lumped inertia of roll servo system
n =	integer number of passes (even no. in Section 4)
p =	Laplace operator with respect to time t
r =	gauge reduction factor
R =	transfer function of the operation: 'record function and reverse its time sequence'
$\mathbf{R}$ =	matrix of transfer functions R, and zeros
s =	Laplace operator with respect to v' (and v where this denotes a variable distance)
S =	speed of pass (at output in the case of the rolling process)
T =	various process time constants and delays (with various suffixes appended)
$\mathbf{u}$ =	vector of process inputs
v' =	variable denoting total distance passed
v =	as v' (except in Section 4 where v denotes a particular value of v')
X =	displacement between process output sensor and process tool
$\mathbf{y}$ =	process output vector
y =	process output; coal thickness, distance between consecutive furrows, according to the process in question
y_{ref} =	reference value of y
†y_i =	input strip gauge
†y_o =	output strip gauge
†y_1 =	screw down displacement
†y_{1d} ≡	demanded y_1
y_{oref} =	reference value of y_o
z =	process disturbance
λ_1 =	stiffness of work rolls and their supports
λ_2 =	yield coefficient of metal strip
Λ =	composite stiffness of strip and rolls
ζ =	damping ratio
ω_0 =	undamped natural frequency
$G'(s), G'_F(j\omega)$ =	transfer function and Fourier transform of weighting function of single-pass closed loop

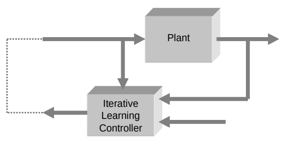
Paper 7307 C, received 31st July 1974

Mr. Edwards is with the Department of Control Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, England, and was formerly with the National Coal Board

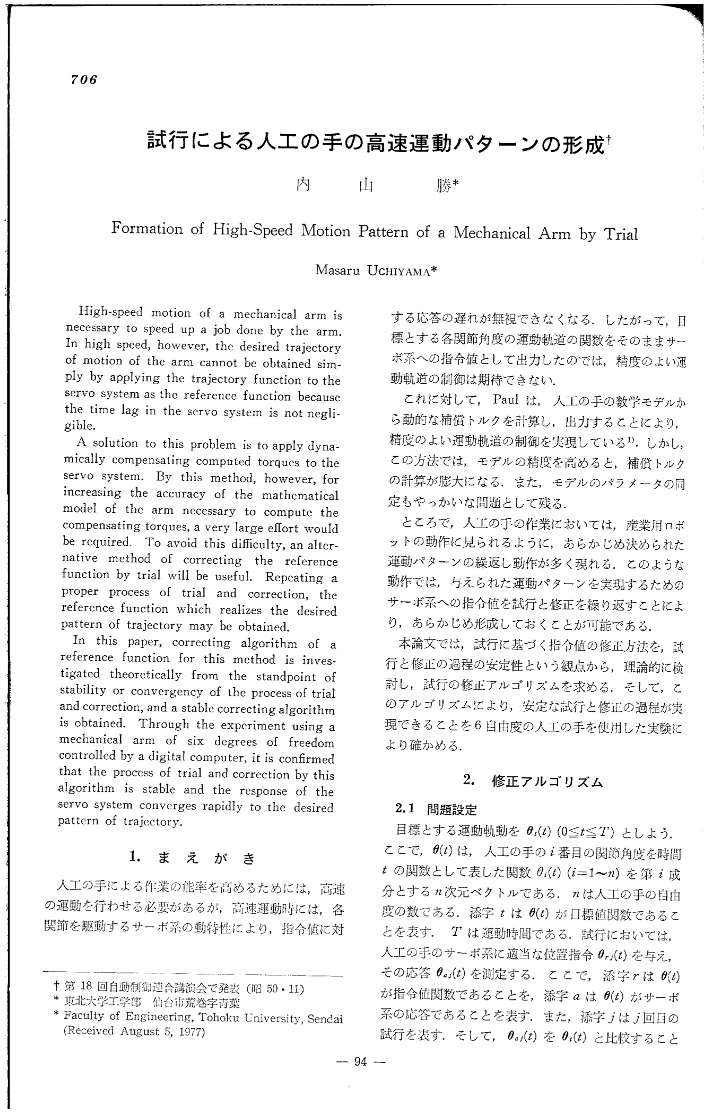
1 Introduction

The rolling of metal strip, the ploughing of agricultural land, the longwall cutting of coal and a variety of machining operations are processes which are similar in

† These symbols are also used to represent the process variables normalised with respect to their time-varying references



First ILC paper- in Japanese (1978)



706

試行による人工の手の高速運動パターンの形成[†]

内　山　　勝*

Formation of High-Speed Motion Pattern of a Mechanical Arm by Trial

Masaru Uchiyama*

High-speed motion of a mechanical arm is necessary to speed up a job done by the arm. In high speed, however, the desired trajectory of motion of the arm cannot be obtained simply by applying the trajectory function to the servo system as the reference function because the time lag in the servo system is not negligible.

A solution to this problem is to apply dynamically compensating computed torques to the servo system. By this method, however, for increasing the accuracy of the mathematical model of the arm necessary to compute the compensating torques, a very large effort would be required. To avoid this difficulty, an alternative method of correcting the reference function by trial will be useful. Repeating a proper process of trial and correction, the reference function which realizes the desired pattern of trajectory may be obtained.

In this paper, correcting algorithm of a reference function for this method is investigated theoretically from the standpoint of stability or convergency of the process of trial and correction, and a stable correcting algorithm is obtained. Through the experiment using a mechanical arm of six degrees of freedom controlled by a digital computer, it is confirmed that the process of trial and correction by this algorithm is stable and the response of the servo system converges rapidly to the desired pattern of trajectory.

1.　ま　え　が　き

人工の手による作業の能率を高めるためには，高速の運動を行わせる必要があるが，高速運動時には，各関節を駆動するサーボ系の動特性により，指令値に対する応答の遅れが無視できなくなる．したがって，目標とする各関節角度の運動軌道の関数をそのままサーボ系への指令値として出力したのでは，精度のよい運動軌道の制御は期待できない．

これに対して，Paul は，人工の手の数学モデルから動的な補償トルクを計算し，出力することにより，精度のよい運動軌道の制御を実現している[1]．しかし，この方法では，モデルの精度を高めると，補償トルクの計算が膨大になる．また，モデルのパラメータの同定もやっかいな問題として残る．

ところで，人工の手の作業においては，産業用ロボットの動作に見られるように，あらかじめ決められた運動パターンの繰返し動作が多く現れる．このような動作では，与えられた運動パターンを実現するためのサーボ系への指令値を試行と修正を繰り返すことにより，あらかじめ形成しておくことが可能である．

本論文では，試行に基づく指令値の修正方法を，試行と修正の過程の安定性という観点から，理論的に検討し，試行の修正アルゴリズムを求める．そして，このアルゴリズムにより，安定な試行と修正の過程が実現できることを6自由度の人工の手を使用した実験により確かめる．

2.　修正アルゴリズム

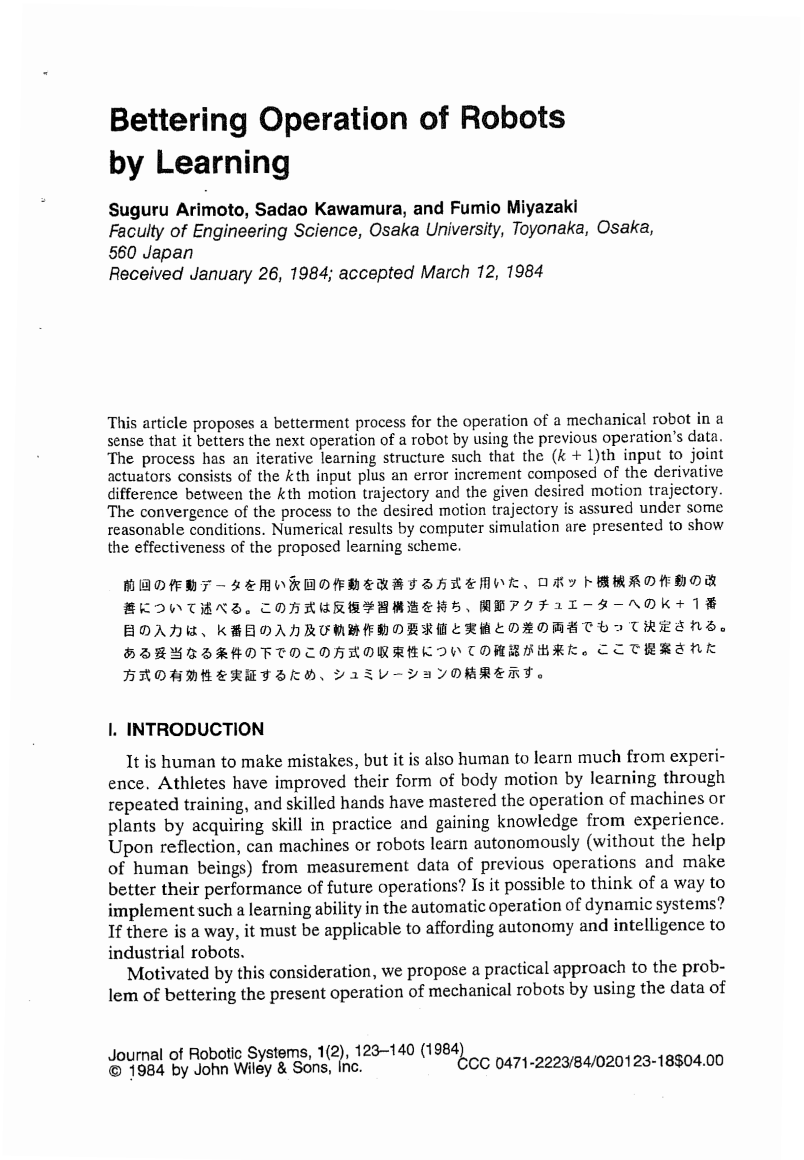
2.1　問題設定

目標とする運動軌動を $\boldsymbol{\theta}_t(t)$ $(0 \leqq t \leqq T)$ としよう．ここで，$\boldsymbol{\theta}(t)$ は，人工の手の i 番目の関節角度を時間 t の関数として表した関数 $\theta_i(t)$ $(i=1 \sim n)$ を第 i 成分とする n 次元ベクトルである．n は人工の手の自由度の数である．添字 t は $\boldsymbol{\theta}(t)$ が目標値関数であることを表す．T は運動時間である．試行においては，人工の手のサーボ系に適当な位置指令 $\boldsymbol{\theta}_{rj}(t)$ を与え，その応答 $\boldsymbol{\theta}_{aj}(t)$ を測定する．ここで，添字 r は $\boldsymbol{\theta}(t)$ が指令値関数であることを，添字 a は $\boldsymbol{\theta}(t)$ がサーボ系の応答であることを表す．また，添字 j は j 回目の試行を表す．そして，$\boldsymbol{\theta}_{aj}(t)$ を $\boldsymbol{\theta}_t(t)$ と比較すること

† 第 18 回自動制御連合講演会で発表（昭 50・11）
* 東北大学工学部　仙台市荒巻字青葉
* Faculty of Engineering, Tohoku University, Sendai
(Received August 5, 1977)

— 94 —

First ILC paper- in English (1984)



Bettering Operation of Robots by Learning

Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki
Faculty of Engineering Science, Osaka University, Toyonaka, Osaka, 560 Japan
Received January 26, 1984; accepted March 12, 1984

This article proposes a betterment process for the operation of a mechanical robot in a sense that it betters the next operation of a robot by using the previous operation's data. The process has an iterative learning structure such that the $(k+1)$th input to joint actuators consists of the kth input plus an error increment composed of the derivative difference between the kth motion trajectory and the given desired motion trajectory. The convergence of the process to the desired motion trajectory is assured under some reasonable conditions. Numerical results by computer simulation are presented to show the effectiveness of the proposed learning scheme.

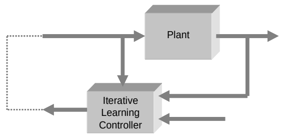
前回の作動データを用い次回の作動を改善する方式を用いた、ロボット機械系の作動の改善について述べる。この方式は反復学習構造を持ち、関節アクチュエーターへのＫ＋１番目の入力は、Ｋ番目の入力及び軌跡作動の要求値と実値との差の両者でもって決定される。ある妥当なる条件の下でのこの方式の収束性についての確認が出来た。ここで提案された方式の有効性を実証するため、シュミレーションの結果を示す。

I. INTRODUCTION

It is human to make mistakes, but it is also human to learn much from experience. Athletes have improved their form of body motion by learning through repeated training, and skilled hands have mastered the operation of machines or plants by acquiring skill in practice and gaining knowledge from experience. Upon reflection, can machines or robots learn autonomously (without the help of human beings) from measurement data of previous operations and make better their performance of future operations? Is it possible to think of a way to implement such a learning ability in the automatic operation of dynamic systems? If there is a way, it must be applicable to affording autonomy and intelligence to industrial robots.

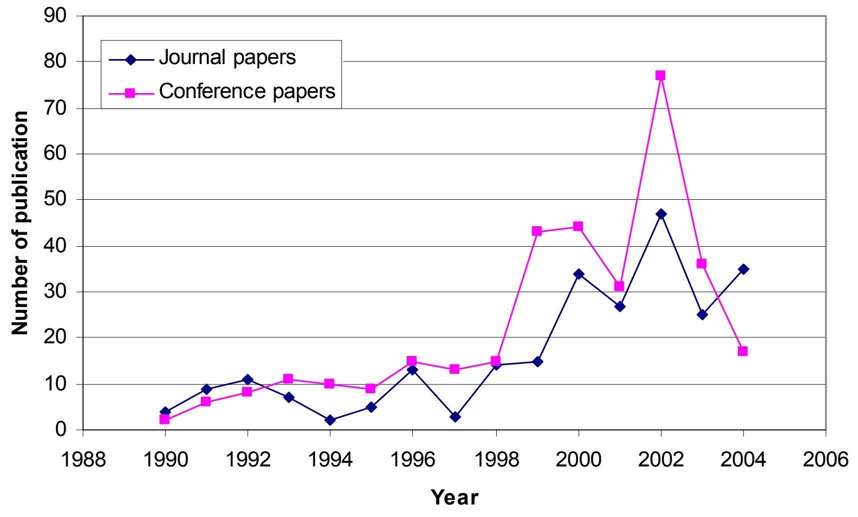
Motivated by this consideration, we propose a practical approach to the problem of bettering the present operation of mechanical robots by using the data of

Journal of Robotic Systems, 1(2), 123–140 (1984)
© 1984 by John Wiley & Sons, Inc. CCC 0471-2223/84/020123-18$04.00



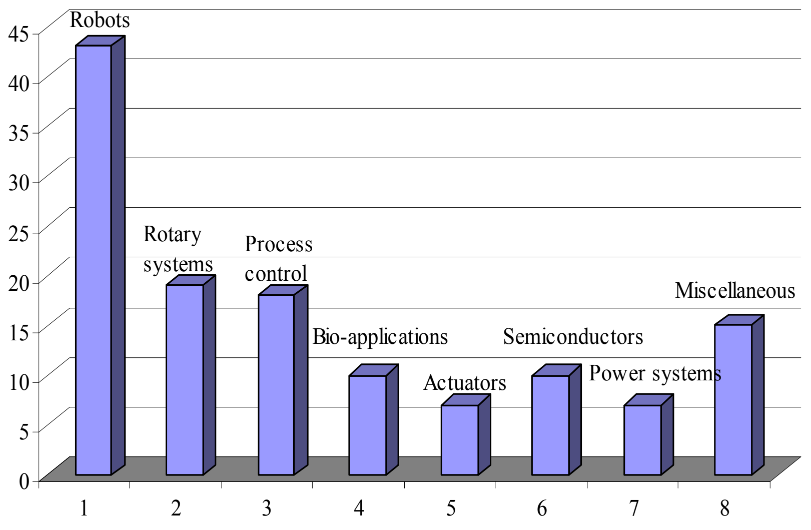
ILC Research History

- ILC has a well-established research history:
  - More than 1000 papers:

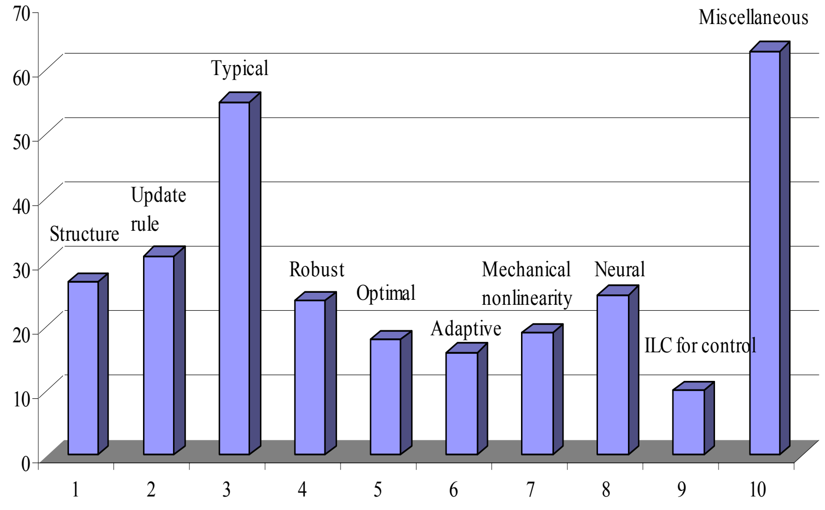


  - At least four monographs.
  - Over 20 Ph.D dissertations.

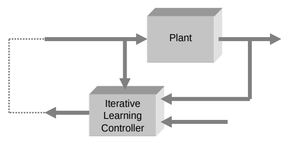
ILC Research History (cont.)



By application areas.

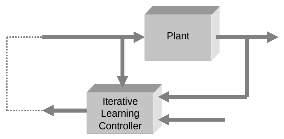


By theoretical areas.

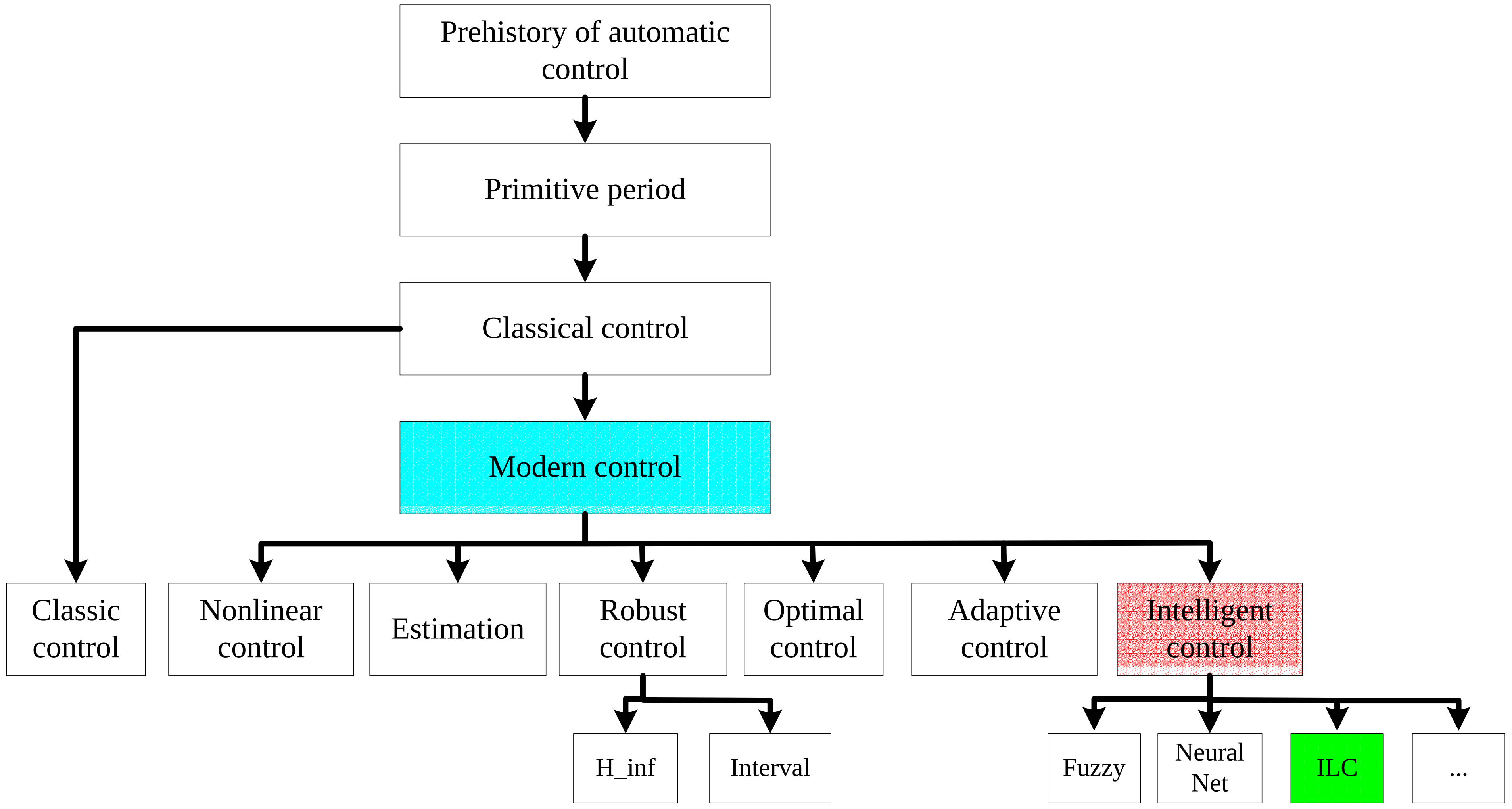


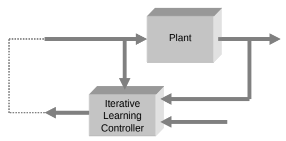
Selected ILC Industrial Applications

- ILC patents in hard disk drive servo:
  - YangQuan Chen's US6,437,936 "*Repeatable runout compensation using a learning algorithm with scheduled parameters.*"
  - YangQuan Chen's US6,563,663 "*Repeatable runout compensation using iterative learning control in a disc storage system.*"
- Robotics:
  - Michael Norrlöf's patent on ABB robots. US2004093119 "*Path correction for an industrial robot.*"
- Gantry motion control:
  - Work by Southampton Sheffield Iterative Learning Control (SSILC) Group.



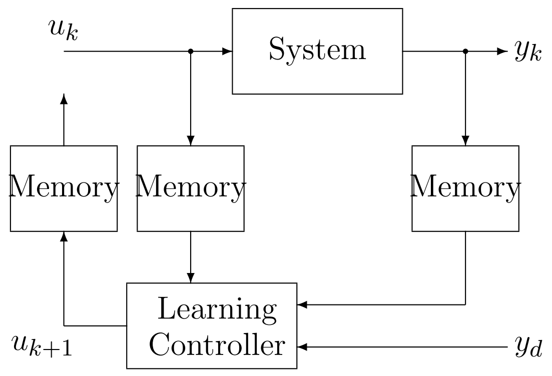
Control Engineering - History and ILC





ILC Problem Formulation

- Standard iterative learning control scheme:



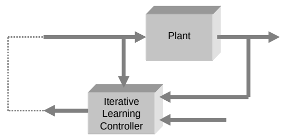
- **Goal**: Find a learning control algorithm

$$u_{k+1}(t) = f_L(\text{previous information})$$

so that for all $t \in [0, t_f]$

$$\lim_{k\to\infty} y_k(t) = y_d(t)$$

- We will consider this problem primarily for discrete-time, linear systems.



Some Learning Control Algorithms

- Arimoto first proposed a learning control algorithm of the form:

$$u_{k+1}(t) = u_k(t) + \Gamma \dot{e}_k(t)$$

  Convergence is assured if $\|I - CB\Gamma\|_i < 1$.

- Arimoto has also considered more general algorithms of the form:

$$u_{k+1} = u_k + \Phi e_k + \Gamma \dot{e}_k + \Psi \int e_k dt$$

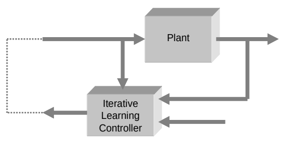
- Various researchers have used gradient methods to optimize the gain G_k in:

$$u_{k+1}(t) = u_k(t) + G_k e_k(t+1)$$

- It is also useful for design to specify the learning control algorithm in the frequency domain, for example:

$$U_{k+1}(s) = L(s)[U_k(s) + aE_k(s)]$$

- Many schemes in the literature can be classified with one of the algorithms given above.



LTI ILC Convergence Conditions

- **Theorem**: For the plant $y_k = T_s u_k$, the linear time-invariant learning control algorithm

$$u_{k+1} = T_u u_k + T_e(y_d - y_k)$$

converges to a fixed point $u^*(t)$ given by

$$u^*(t) = (I - T_u + T_e T_s)^{-1} T_e y_d(t)$$

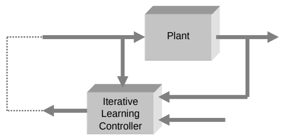
with a final error

$$e^*(t) = \lim_{k\to\infty}(y_k - y_d) = (I - T_s(I - T_u + T_e T_s)^{-1} T_e) y_d(t)$$

defined on the interval (t_0, t_f) if

$$\|T_u - T_e T_s\|_i < 1$$

- **Observation**:
  - If $T_u = I$ then $\|e^*(t)\| = 0$ for all $t \in [t_o, t_f]$.
  - Otherwise the error will be non-zero.



LTI Learning Control - Nature of the Solution

- **Question**: Given T_s, how do we pick T_u and T_e to make the final error $e^*(t)$ as "small" as possible, for the general linear ILC algorithm:

$$u_{k+1}(t) = T_u u_k(t) + T_e(y_d(t) - y_k(t))$$

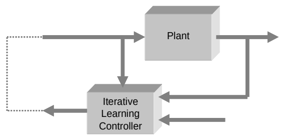
- **Answer**: Let T_n^* solve the problem:

$$\min_{T_n} \|(I - T_s T_n) y_d\|$$

  It turns out that we can specify T_u and T_e in terms of T_n^* and the resulting learning controller converges to an optimal system input given by:

$$u^*(t) = T_n^* y_d(t)$$

- **Conclusion**:The essential effect of a properly designed learning controller is to produce the output of the best possible inverse of the system in the direction of y_d.

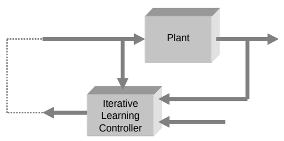


LTI ILC - Solution Details

- **OPT1**: Let $u_k \in \mathbf{U}$, $y_d, y_k \in \mathbf{Y}$ and $T_s, T_u, T_e \in \mathbf{X}$. Then given y_d and T_s, find T_u^* and T_e^* that solve
$$\min_{T_u, T_e \in \mathbf{X}} \|(I - T_s(I - T_u + T_e T_s)^{-1} T_e) y_d\|$$
subject to $\|T_u - T_e T_s\|_i < 1$.

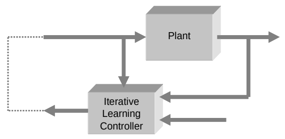
- **OPT2**: Let $y_d \in \mathbf{Y}$ and let $T_n, T_s \in \mathbf{X}$. Then given y_d and T_s, find T_n^* that solve
$$\min_{T_n \in \mathbf{X}} \|(I - T_s T_n) y_d\|$$

- **Theorem**: Let T_n^* be the solution of OPT2. Factor $T_n^* = T_m^* T_e^*$ where $T_m^{*^{-1}} \in \mathbf{X}$ and $\|I - T_m^{*^{-1}}\|_i < 1$. Define $T_u^* = I - T_m^{*^{-1}} + T_e^* T_s$. Then T_u^* and T_e^* are the solution of OPT1.
- If we plug these into the expression for the fixed-point of the input to the system we find:
$$u^*(t) = T_n^* y_d(t)$$
- Note: The factorization in the Theorem can always be done, with the result that $u^*(t) = T_n^* y_d(t)$.



Outline

- Introduction
  - Control System Design: Motivation for ILC
  - Iterative Learning Control: The Basic Idea
  - Some Comments on the History of ILC
  - ILC Problem Formulation
- **The "Supervector" Notation**
- **The w-Transform: "z-Operator" Along the Repetition Axis**
- **ILC as a MIMO Control System**
  - **Repetition-Domain Poles**
  - **Repetition-Domain Internal Model Principle**
- The Complete Framework
  - Repetition-Varying Inputs and Disturbances
  - Plant Model Variation Along the Repetition Axis



ILC as a Two-Dimensional Process

- Suppose the plant is a scalar discrete-time dynamical system, described as:

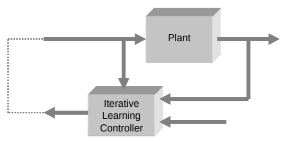
$$y_k(t+1) \; = \; f_S[y_k(t), u_k(t), t]$$

  where
  - k denotes a trial (or execution, repetition, pass, etc.).
  - $t \in [0, N]$ denotes time (integer-valued).
  - $y_k(0) = y_d(0) = y_0$ for all k.

- Use a general form of a typical ILC algorithm for a system with relative degree one:

$$u_{k+1}(t) \; = \; f_L[u_k(t), e_k(t+1), k]$$

  where
  - $e_k(t) = y_d(t) - y_k(t)$ is the error on trial k.
  - $y_d(t)$ is a desired output signal.



ILC as a Two-Dimensional Process (cont.)

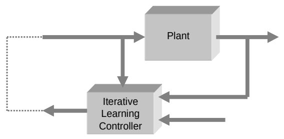
- Combine the plant equation with the ILC update rule to get:

$$y_{k+1}(t+1) \;=\; f_S[y_k(t), u_{k+1}(t), t] = f_S[f_L[y_k(t), u_k(t), e_k(t+1), k], t]$$

- Changing the notation slightly we get:

$$y(k+1, t+1) = f[y_k(t), u(k,t), e(k, t+1), k, t]$$

- Clearly this is a 2-D system:
  - Dynamic equation indexed by two variables: k and t.
  - k defines the repetition domain (Longman/Phan terminology).
  - t is the normal time-domain variable.
- But, ILC differs from a complete 2-D system design problem:
  - One of the dimensions (time) is a finite, fixed interval, thus convergence in that direction (traditional stability) is always assured for linear systems.
  - In the ILC problem we admit non-causal processing in one dimension (time) but not in the other (repetition).
- We can exploit these points to turn the 2-D problem into a 1-D problem.



The "Supervector" Framework of ILC

- Consider an SISO, LTI discrete-time plant with relative degree m:

$$Y(z) = H(z)U(z) = (h_m z^{-m} + h_{m+1} z^{-(m+1)} + h_{m+2} z^{-(m+2)} + \cdots)U(z)$$

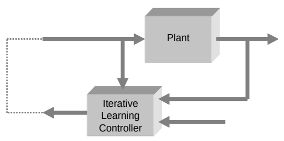
- By "lifting" along the time axis, for each trial k define:

$$\begin{aligned} U_k &= [u_k(0), u_k(1), \cdots, u_k(N-1)]^T \\ Y_k &= [y_k(m), y_k(m+1), \cdots, y_k(m+N-1)]^T \\ Y_d &= [y_d(m), y_d(m+2), \cdots, y_d(m+N-1)]^T \end{aligned}$$

- Thus the linear plant can be described by $Y_k = H_p U_k$ where:

$$H_p = \begin{bmatrix} h_1 & 0 & 0 & \dots & 0 \\ h_2 & h_1 & 0 & \dots & 0 \\ h_3 & h_2 & h_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_N & h_{N-1} & h_{N-2} & \dots & h_1 \end{bmatrix}$$

- The lower triangular matrix H_p is formed using the system's Markov parameters.
- Notice the non-causal shift ahead in forming the vectors U_k and Y_k.



The "Supervector" Framework of ILC (cont.)

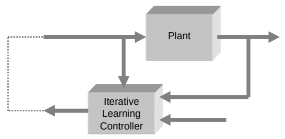
- For the linear, time-varying case, suppose we have the plant given by:

$$\begin{aligned} x_k(t+1) &= A(t)x_k(t) + B(t)u_k(t) \\ y_k(t) &= C(t)x_k(t) + D(t)u_k(t) \end{aligned}$$

Then the same notation again results in $Y_k = H_p U_k$, where now:

$$H_p = \begin{bmatrix} h_{m,0} & 0 & 0 & \dots & 0 \\ h_{m+1,0} & h_{m,1} & 0 & \dots & 0 \\ h_{m+2,0} & h_{m+1,1} & h_{m,2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{m+N-1,0} & h_{m+N-2,1} & h_{m+N-3,2} & \dots & h_{m,N-1} \end{bmatrix}$$

- The lifting operation over a finite interval allows us to:
  - Represent our dynamical system in R^1 into a static system in R^N.



The Update Law Using Supervector Notation

- Suppose we have a simple "Arimoto-style" ILC update equation with a constant gain γ:
  - In our R^1 representation, we write:

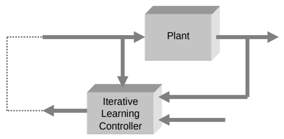
$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1)$$

  - In our R^N representation, we write:

$$U_{k+1} = U_k + \Gamma E_k$$

where

$$\Gamma = \begin{bmatrix} \gamma & 0 & 0 & \dots & 0 \\ 0 & \gamma & 0 & \dots & 0 \\ 0 & 0 & \gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \gamma \end{bmatrix}$$



The Update Law Using Supervector Notation (cont.)

- Suppose we filter with an LTI filter during the ILC update:
  - In our R^1 representation we would have the form:

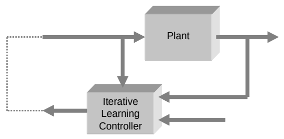
$$u_{k+1}(t) = u_k(t) + L(z)e_k(t+1)$$

  - In our R^N representation we would have the form:

$$U_{k+1} = U_k + LE_k$$

where L is a Topelitz matrix of the Markov parameters of $L(z)$, given, in the case of a "causal," LTI update law, by:

$$L = \begin{bmatrix} L_m & 0 & 0 & \dots & 0 \\ L_{m+1} & L_m & 0 & \dots & 0 \\ L_{m+2} & L_{m+1} & L_m & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{m+N-1} & L_{m+N-2} & L_{m+N-3} & \dots & L_m \end{bmatrix}$$



The Update Law Using Supervector Notation (cont.)

- We may similarly consider time-varying and noncausal filters in the ILC update law:

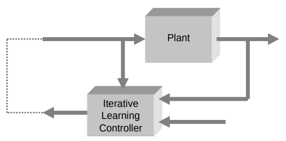
$$U_{k+1} = U_k + LE_k$$

- A causal (in time), time-varying filter in the ILC update law might look like, for example:

$$L = \begin{bmatrix} n_{1,0} & 0 & 0 & \dots & 0 \\ n_{2,0} & n_{1,1} & 0 & \dots & 0 \\ n_{3,0} & n_{2,1} & n_{1,2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n_{N,0} & n_{N-1,1} & n_{N-2,2} & \dots & n_{1,N-1} \end{bmatrix}$$

- A non-causal (in time), time-invariant averaging filter in the ILC update law might look like, for example:

$$L = \begin{bmatrix} K & K & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & K & K & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & K & K & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & K & K & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & K & K \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & K \end{bmatrix}$$



The Update Law Using Supervector Notation (cont.)

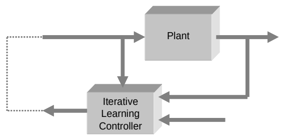
- The supervector notation can also be applied to other ILC update schemes. For example:
  - The Q-filter often introduced for stability (along the iteration domain) has the R^1 representation:

$$u_{k+1}(t) = Q(z)(u_k(t) + L(z)e_k(t+1))$$

  - The equivalent R^N representation is:

$$U_{k+1} = Q(U_k + LE_k)$$

where Q is a Toeplitz matrix formed using the Markov parameters of the filter $Q(z)$.

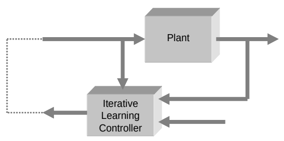


The ILC Design Problem

- The design of an ILC controller can be thought of as the selection of the matrix L:
  - For a causal ILC updating law, L will be in lower-triangular Toeplitz form.
  - For a noncausal ILC updating law, L will be in upper-triangular Toeplitz form.
  - For the popular zero-phase learning filter, L will be in a symmetrical band diagonal form.
  - L can also be fully populated.
- Motivated by these comments, we will refer to the "causal" and "non-causal" elements of a general matrix Γ as follows:

$$\Gamma = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & \cdots & \gamma_{1N} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & \text{noncausal} & \gamma_{2N} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & \cdots & \gamma_{3N} \\ \vdots & \text{causal} & \vdots & \ddots & \vdots \\ \gamma_{N1} & \gamma_{N2} & \gamma_{N3} & \cdots & \gamma_{NN} \end{bmatrix}$$

The diagonal elements of Γ are referred to as "Arimoto" gains.



w-Transform: the "z-Operator" in the Iteration Domain

- Introduce a new shift variable, w, with the property that, for each fixed integer t:

$$w^{-1}u_k(t) = u_{k-1}(t)$$

- For a scalar $x_k(t)$, combining the lifting operation to get the supervector X_k with the shift operation gives what we call the w-transform of $x_k(t)$, which we denote by $X(w)$
- Then the ILC update algorithm:

$$u_{k+1}(t) = u_k(t) + L(z)e_k(t+1)$$

which, using our supervector notation, can be written as $U_{k+1} = U_k + LE_k$ can also be written as:

$$wU(w) = U(w) + LE(w)$$

where $U(w)$ and $E(w)$ are the w-transforms of U_k and E_k, respectively.

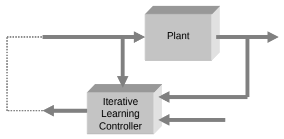
- Note that we can also write this as

$$E(w) = C(w)U(w)$$

where

$$C(w) = \frac{1}{(w-1)}L$$

.

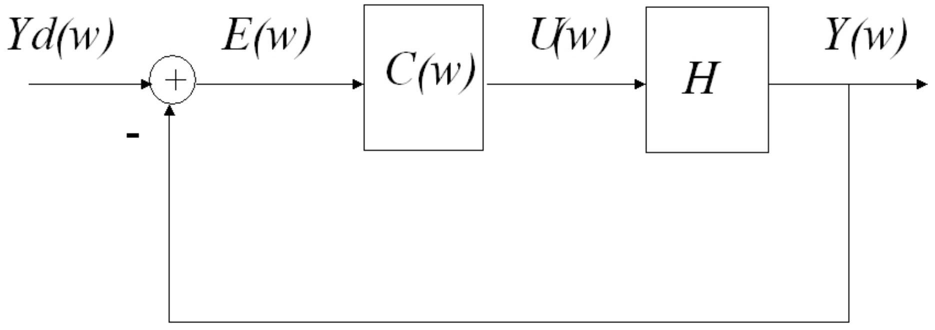


ILC as a MIMO Control System

- The term

$$C(w) = \frac{1}{(w-1)}L$$

is effectively the controller of the system (in the repetition domain). This can be depicted as:



Higher-Order ILC in the Iteration Domain

- We can use these ideas to develop more general expressions ILC algorithms.

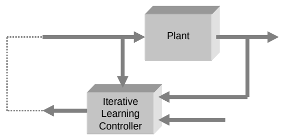
- For example, a "higher-order" ILC algorithm could have the form:

$$u_{k+1}(t) = k_1 u_k(t) + k_2 u_{k-1}(t) + \gamma e_k(t+1)$$

  which corresponds to:

$$C(w) = \frac{\gamma w}{w^2 - k_1 w - k_2}$$

- Next we show how to extend these notions to develop an algebraic (matrix fraction) description of the ILC problem.



A Matrix Fraction Formulation

- Suppose we consider a more general ILC update equation given by (for relative degree $m = 1$):

$$\begin{aligned} u_{k+1}(t) &= \bar{D}_n(z)u_k(t) + \bar{D}_{n-1}(z)u_{k-1}(t) + \cdots + \bar{D}_1(z)u_{k-n+1}(t) + \bar{D}_0(z)u_{k-n}(t) \\ &\quad + N_n(z)e_k(t+1 + N_{n-1}(z)e_{k-1}(t+1 + \cdots + N_1(z)e_{k-n+1}(t+1) + N_0(z)e_{k-n}(t+1) \end{aligned}$$

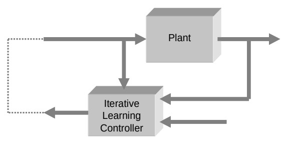
which has the supervector expression

$$\begin{aligned} U_{k+1} &= \bar{D}_n U_k + \bar{D}_{n-1}U_{k-1} + \cdots + \bar{D}_1 U_{k-n+1} + \bar{D}_0 U_{k-n} \\ &\quad + N_n E_k + N_{n-1}E_{k-1} + \cdots + N_1 E_{k-n+1} + N_0 E_{k-n} \end{aligned}$$

- Aside: note that there are a couple of variations on the theme that people sometimes consider:

  – $U_{k+1} = U_k + LE_{k+1}$
  – $U_{k+1} = U_k + L_1 E_k + L_0 E_{k+1}$

  These can be accomodated by adding a term $N_{n+1}E_{k+1}$ in the expression above, resulting in the so-called "current iteration feedback," or CITE.



A Matrix Fraction Formulation (cont.)

- Applying the shift variable w we get:

$$\bar{D}_c(w)U(w) = N_c(w)E(w)$$

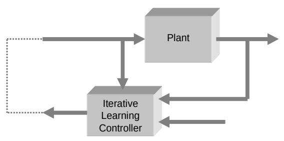
where

$$\begin{aligned} \bar{D}_c(w) &= Iw^{n+1} - \bar{D}_{n-1}w^n - \cdots - \bar{D}_1 w - \bar{D}_0 \\ N_c(w) &= N_n w^n + N_{n-1}w^{n-1} + \cdots + N_1 w + N_0 \end{aligned}$$

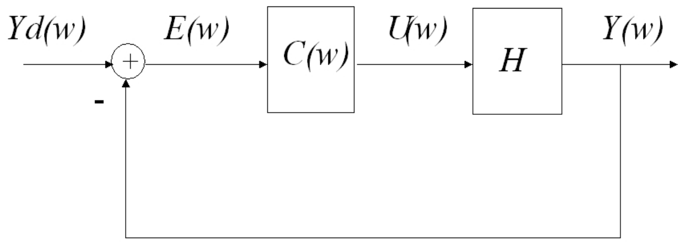
- This can be written in a matrix fraction as $U(w) = C(w)E(w)$ where:

$$C(w) = \bar{D}_c^{-1}(w)N_c(w)$$

- Thus, through the addition of higher-order terms in the update algorithm, the ILC problem has been converted from a static multivariable representation to a dynamic (in the repetition domain) multivariable representation.
- Note that we will always get a linear, time-invariant system like this, even if the actual plant is time-varying.
- Also, because $\bar{D}_c(w)$ is of degree $n+1$ and $N_c(w)$ is of degree n, we have relative degree one in the repetition-domain, unless some of the gain matrices are set to zero.



ILC Convergence via Repetition-Domain Poles



- From the figure we see that in the repetition-domain the closed-loop dynamics are defined by:

$$\begin{aligned} G_{cl}(w) &= H_p[I + C(w)H_p]^{-1}C(w) \\ &= H_p[\bar{D}_c(w) + N_c(w)H_p]^{-1}N_c(w) \end{aligned}$$

- Thus the ILC algorithm will converge (i.e., $E_k \rightarrow$ a constant) if G_{cl} is stable.
- Determining the stability of this feedback system may not be trivial:
  - It is a multivariable feedback system of dimension N, where N could be very large.
  - But, the problem is simplified due to the fact that the plant H_p is a constant, lower-triangular matrix.

Repetition-Domain Internal Model Principle

- Because Y_d is a constant and our "plant" is type zero (e.g., H_p is a constant matrix), the internal model principle applied in the repetition domain requires that $C(w)$ should have an integrator effect to cause $E_k \rightarrow 0$.
- Thus, we modify the ILC update algorithm as:

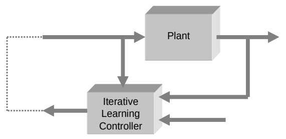
$$\begin{aligned} U_{k+1} &= (I - D_{n-1})U_k + (D_{n-1} - D_{n-2})U_{k-1} + \cdots \\ &\quad +(D_2 - D_1)U_{k-n+2} + (D_1 - D_0)U_{k-n+1} + D_0 U_{k-n} \\ &\quad +N_n E_k + N_{n-1}E_{k-1} + \cdots + N_1 E_{k-n+1} + N_0 E_{k-n} \end{aligned}$$

- Taking the "w-transform" of the ILC update equation, combining terms, and simplifying gives:

$$(w-1)D_c(w)U(w) = N_c(w)E(w)$$

where

$$\begin{aligned} D_c(w) &= w^n + D_{n-1}w^{n-1} + \cdots + D_1 w + D_0 \\ N_c(w) &= N_n w^n + N_{n-1}w^{n-1} + \cdots + N_1 w + N_0 \end{aligned}$$



Repetition-Domain Internal Model Principle (cont.)

- This can also be written in a matrix fraction as:

$$U(w) = C(w)E(w)$$

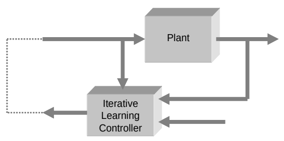
  but where we now have:

$$C(w) = (w-1)^{-1}D_c^{-1}(w)N_c(w)$$

- For this update law the repetition-domain closed-loop dynamics become:

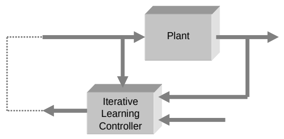
$$\begin{aligned} G_{cl}(w) &= H\left(I + \frac{I}{(w-1)}C(w)H\right)^{-1}\frac{I}{(w-1)}C(w), \\ &= H[(w-1)D_c(w) + N_c(w)H]^{-1}N_c(w) \end{aligned}$$

- Thus, we now have an integrator in the feedback loop (a discrete integrator, in the repetition domain) and, applying the final value theorem to G_{cl}, we get $E_k \to 0$ as long as the ILC algorithm converges (i.e., as long as G_{cl} is stable).



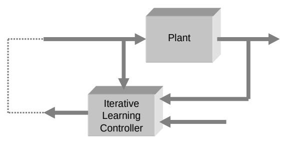
Outline

- Introduction
  - Control System Design: Motivation for ILC
  - Iterative Learning Control: The Basic Idea
  - Some Comments on the History of ILC
  - ILC Problem Formulation
- The "Supervector" Notation
- The w-Transform: "z-Operator" Along the Repetition Axis
- ILC as a MIMO Control System
  - Repetition-Domain Poles
  - Repetition-Domain Internal Model Principle
- **The Complete Framework**
  - **Repetition-Varying Inputs and Disturbances**
  - **Plant Model Variation Along the Repetition Axis**



Higher-Order ILC in the Iteration Domain, Revisited

- A key feature of our matrix fraction, algebraic framework is that it assumes use of higher-order ILC.
- At the '02 IFAC World Congress a special session explored the value that could be obtained from such algorithms:
  - One possible benefit could be due to more freedom in placing the poles (in the w-plane).
  - It has been suggested in the literature that such schemes can give faster convergence.
  - However, we can show dead-beat convergence using any order ILC. Thus, higher-order ILC can be no faster than first-order.
- One conclusion from the '02 IFAC special sessions is that higher-order ILC is primarily beneficial when there is repetition-domain uncertainty.
- Several such possibilities arise:
  - Iteration-to-iteration reference variation.
  - Iteration-to-iteration disturbances and noise.
  - Plant model variation from repetition-to-repetition.
- The matrix fraction, or algebraic, approach can help in these cases.



Iteration-Varying Disturbances

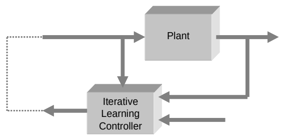
- In ILC, it is assumed that desired trajectory $y_d(t)$ and external disturbance are invariant with respect to iterations.
- When these assumptions are not valid, conventional integral-type, first-order ILC will no longer work well.
- In such a case, ILC schemes that are higher-order along the iteration direction will help.
- Consider a stable plant

$$H_a(z) = \frac{z - 0.8}{(z - 0.55)(z - 0.75)}$$

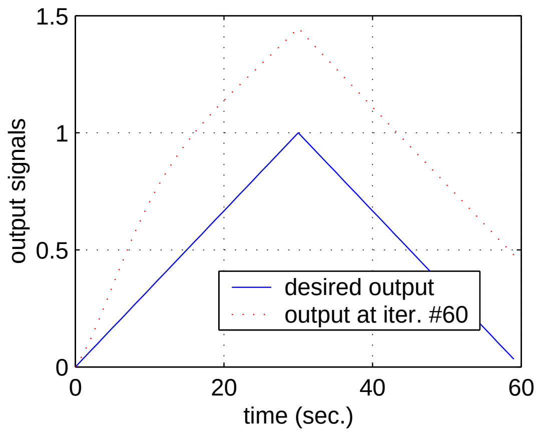
- Let the plant be subject to an additive output disturbance

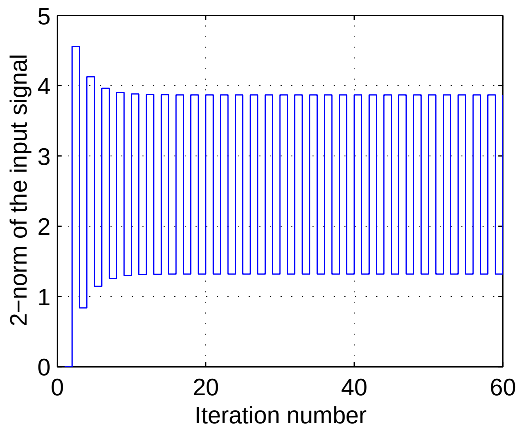
$$d(k, t) = 0.01(-1)^{k-1}$$

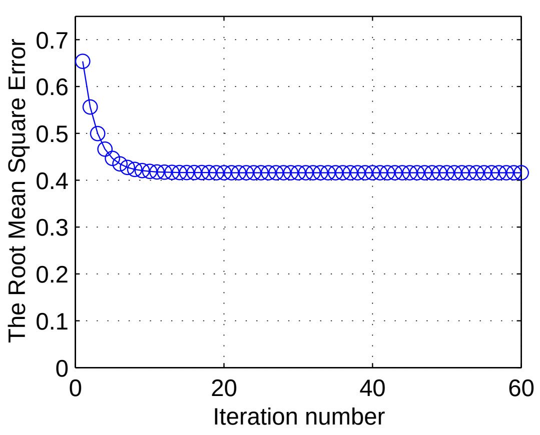
- This is an iteration-varying, alternating disturbance. If the iteration number k is odd, the disturbance is a positive constant in iteration k while when k is even, the disturbance jumps to a negative constant.
- In the simulation, we wish to track a ramp up and down on a finite interval.

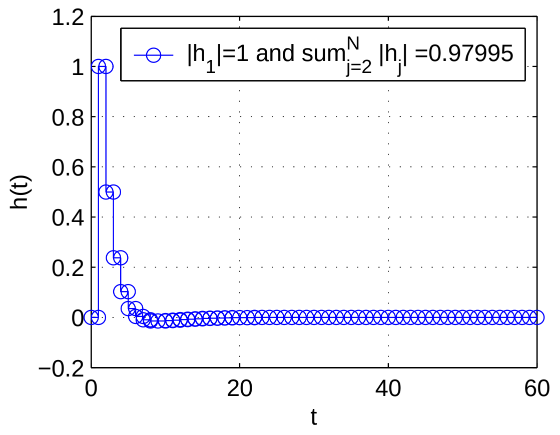


Example: First-Order ILC

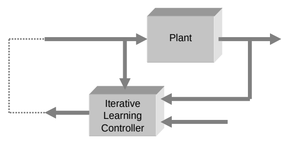




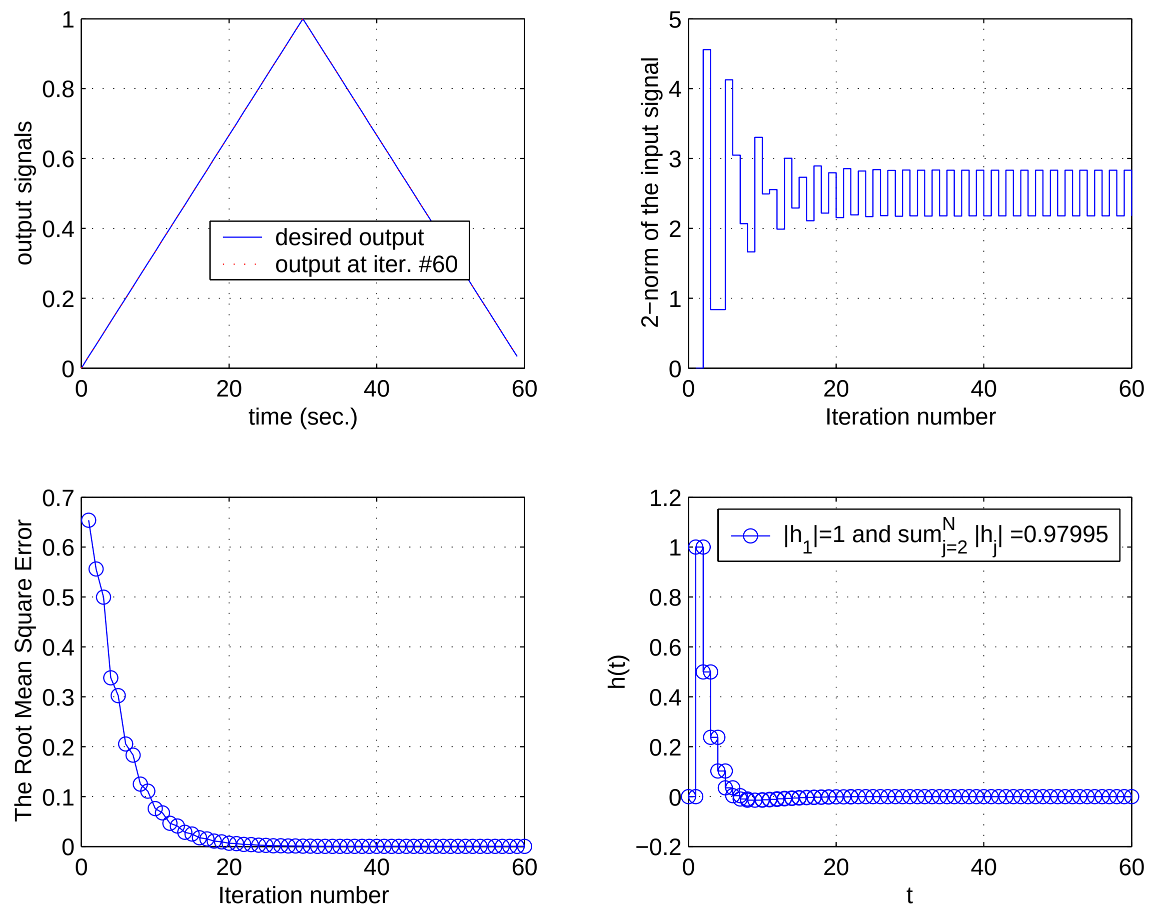




$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1), \gamma = 0.9 \Rightarrow C(w) = \frac{1}{(w-1)}L$$



Example: Second-Order, Internal Model ILC



$$u_{k+1}(t) = u_{k-1}(t) + \gamma e_{k-1}(t+1) \text{ with } \gamma = 0.9 \Rightarrow C(w) = \frac{1}{(w^2-1)} L$$

Example 2 - Iteration-Domain Ramp Disturbance

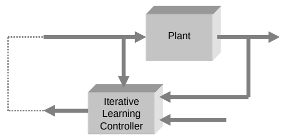
- Consider a stable plant

$$H_a(z) = \frac{z - 0.8}{(z - 0.55)(z - 0.75)}.$$

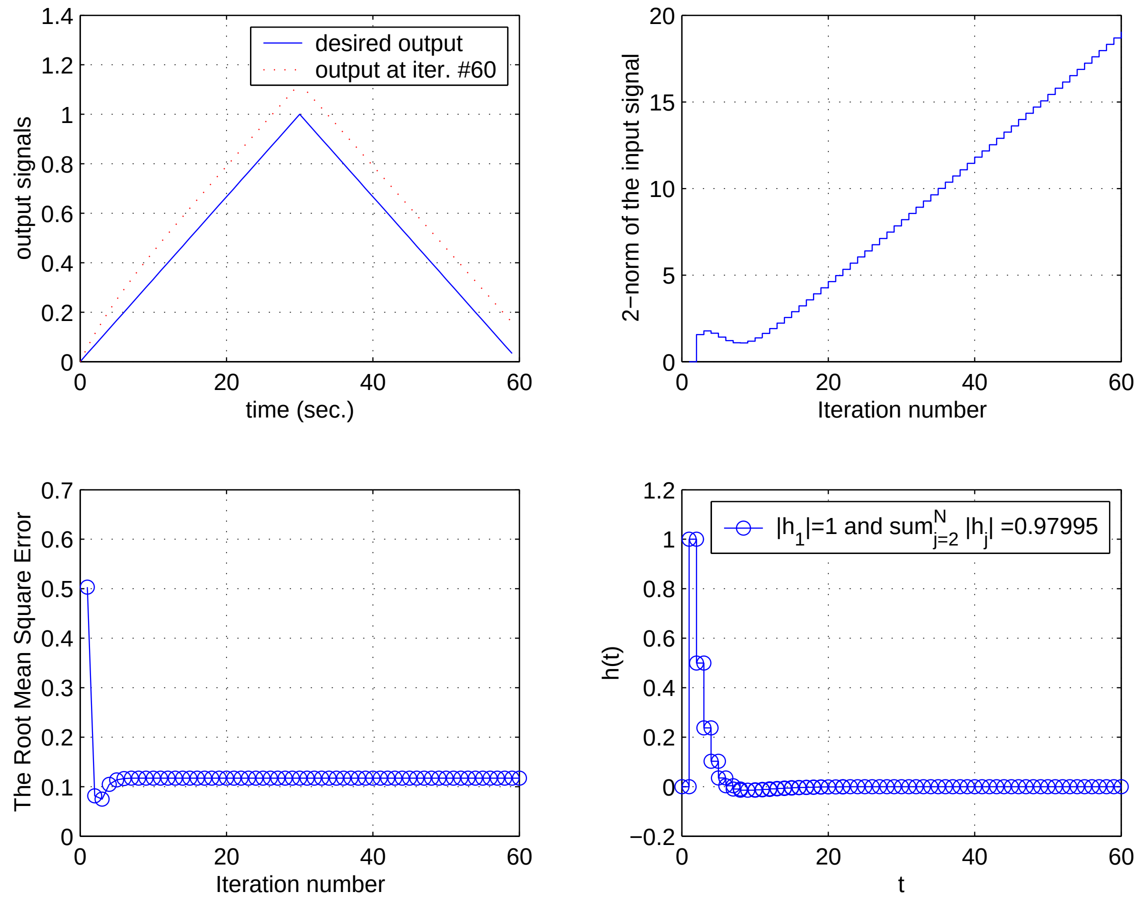
- Assume that $y_d(t)$ does not vary w.r.t. iterations.
- However, we add a disturbance $d(k, t)$ at the output $y_k(t)$.
- In iteration k, the disturbance is a constant w.r.t. time but its value is proportional to k. Thus

$$d(k, t) = c_0 k$$

- In the simulation, we set $c_0 = 0.01$.

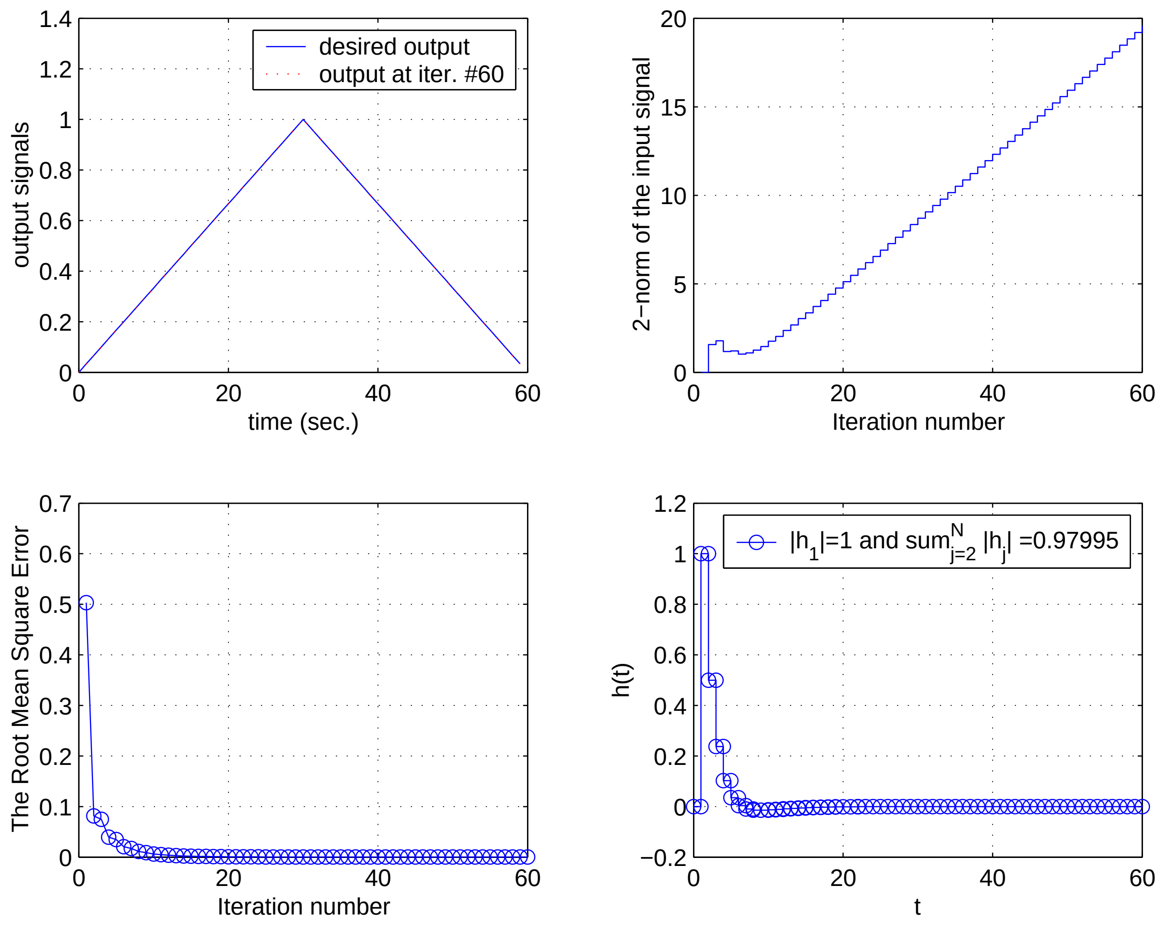


Example 2: First-Order ILC



$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1), \gamma = 0.9$$

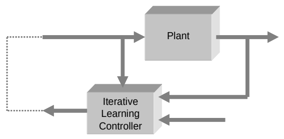
Example 2: Second-Order, Internal Model ILC



$$u_{k+1}(t) = 2u_k(t) - u_{k-1}(t) + \gamma(2e_k(t+1) - e_{k-1}(t+1)), \gamma = 0.9$$

A Complete Design Framework

- We have presented several important facts about ILC:
  - The supervector notation lets us write the ILC system as a matrix fraction, introducing an algebraic framework.
  - In this framework we are able to discuss convergence in terms of pole in the iteration-domain.
  - In this framework we can consider rejection of iteration-dependent disturbances and noise as well as the tracking of iteration-dependent reference signals (by virtue of the internal model principle).
- In the same line of thought, we can next introduce the idea of iteration-varying models.



Iteration-Varying Plants

- Can view the classic multi-pass (Owens and Edwards) and linear repetitive systems (Owens and Rogers) as a generalization of the static MIMO system $Y_k = H_p U_k$ into a dynamic (in iteration) MIMO system, so that

$$Y_{k+1} = A_0 Y_k + B_0 U_k$$

becomes

$$H(w) = (wI - A_0)^{-1} B_0$$

- Introduce iteration-varying plant uncertainty, so the static MIMO system $Y_k = H_p U_k$ becomes a dynamic (in iteration) and uncertain MIMO system, such as

$$H_p = H_0(I + \Delta H)$$

or

$$H_p \in [\underline{H}, \overline{H}]$$

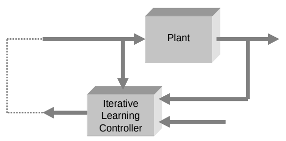
or

$$H_p = H_0 + \Delta H(w)$$

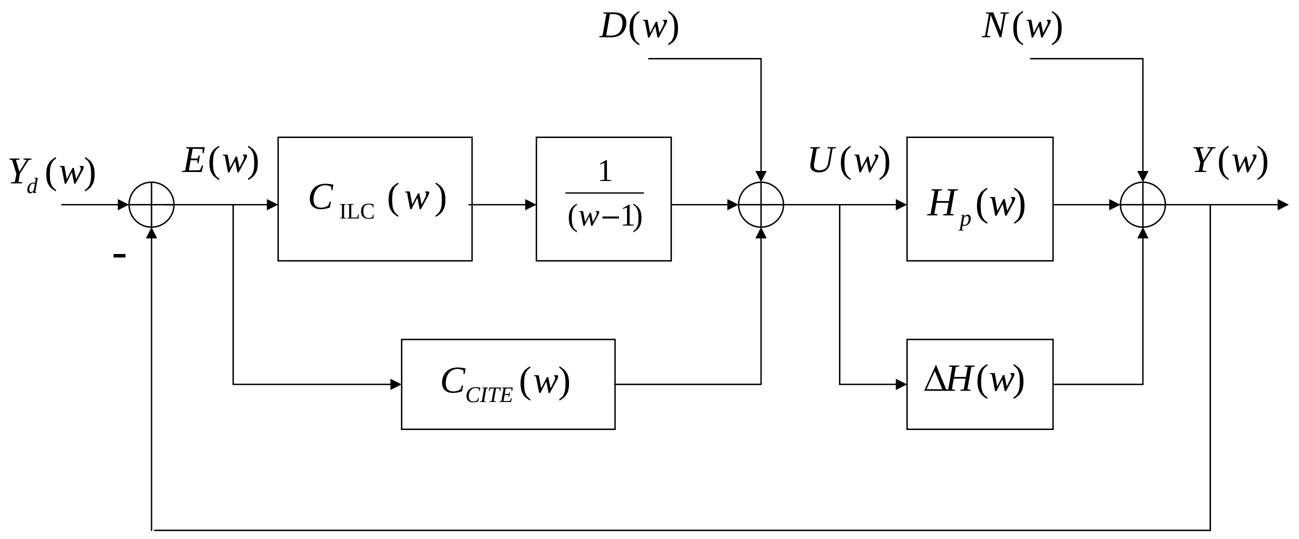
or

$$H_p(w) = H_0(w)(I + \Delta H(w))$$

$$\cdots \text{ etc. } \cdots$$



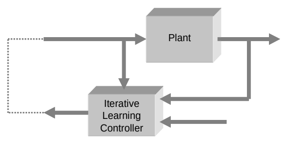
Complete Framework



- $Y_d(w)$, $D(w)$ and $N(w)$ describe, respectively, the iteration-varying reference, disturbance, and noise signals. $H_p(w)$ denotes the (possibly iteration varying) plant.
- $\Delta H_p(w)$ represents the uncertainty in plant model, which may also be iteration-dependent.
- $C_{\text{ILC}}(w)$ denotes the ILC update law.
- $C_{\text{CITE}}(w)$ denotes any current iteration feedback that might be employed.
- The term $\frac{1}{(w-1)}$ denotes the natural one-iteration delay inherent in ILC.

Categorization of Problems

Y_d	D	N	C	H	
$Y_d(z)$	0	0	$\Gamma(w-1)^{-1}$	H_p	Classical Arimoto algorithm
$Y_d(z)$	0	0	$\Gamma(w-1)^{-1}$	$H_p(w)$	Owens' multipass problem
$Y_d(z)$	$D(w)$	0	$C(w)$	H_p	General (higher-order) problem
$Y_d(w)$	$D(w)$	0	$C(w)$	H_p	General (higher-order) problem
$Y_d(w)$	$D(w)$	$N(w)$	$C(w)$	$H_p(w)+\Delta(w)$	**Most general problem**
$Y_d(z)$	$D(z)$	0	$C(w)$	$H(w)+\Delta(w)$	Frequency-domain uncertainty
$Y_d(z)$	0	$w(t), v(t)$	$\Gamma(w-1)^{-1}$	H_p	Least quadratic ILC
$Y_d(z)$	0	$w(t), v(t)$	$C(w)$	H_p	Stochastic ILC (general)
$Y_d(z)$	0	0	$\Gamma(w-1)^{-1}$	H^I	Interval ILC
$Y_d(z)$	$D(z)$	$w(t), v(t)$	$C(w)$	$H(z)+\Delta H(z)$	Time-domain H_∞ problem
$Y_d(z)$	$D(w)$	$w(k,t), v(k,t)$	$C(w)$	$H(w)+\Delta H(w)$	Iteration-domain H_∞ ILC
$Y_d(z)$	0	$w(k,t), v(k,t)$	$\Gamma(k)$	$H_p+\Delta H(k)$	Iteration-varying uncertainty and control
$Y_d(z)$	0	$\widetilde{H}$	Γ	H_p	Intermittent measurement problem
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$



Outline

- Introduction
  - Control System Design: Motivation for ILC
  - Iterative Learning Control: The Basic Idea
  - Some Comments on the History of ILC
- The “Supervector” Notation
- The w-Transform: “z-Operator” Along the Repetition Axis
- ILC as a MIMO Control System
  - Repetition-Domain Poles
  - Repetition-Domain Internal Model Principle
- The Complete Framework
  - Repetition-Varying Inputs and Disturbances
  - Plant Model Variation Along the Repetition Axis