An Introduction to Iterative Learning Control Theory

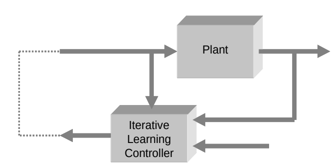
Presenter:

Kevin L. Moore – Colorado School of Mines

Contributors:

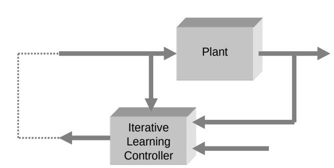
YangQuan Chen - Utah State University
Hyo-sung Ahn - Utah State University

CSM EGES 504/604A Colloquium
Golden, CO — 24 Jamuary 2006

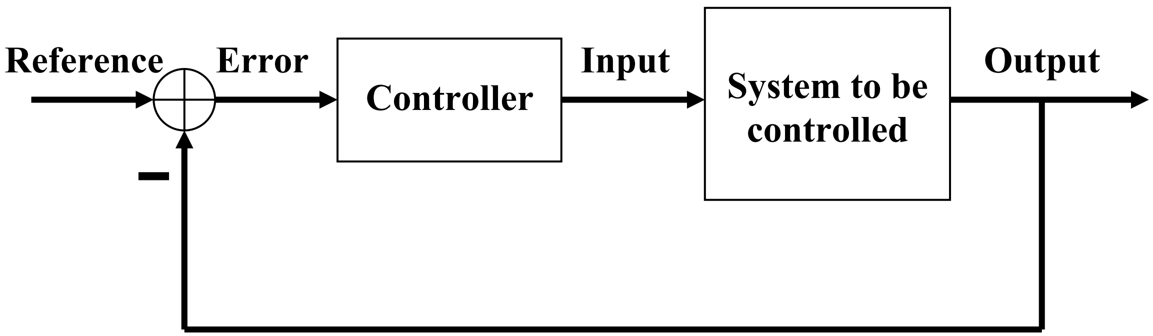


Outline

- 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



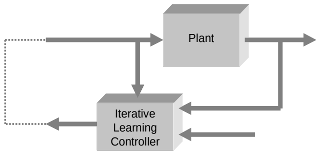
Control Design Problem



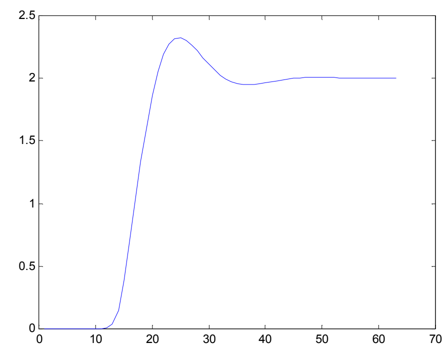
Given: System to be controlled.

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

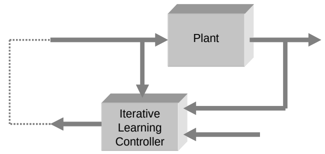


- Transient response design is hard:

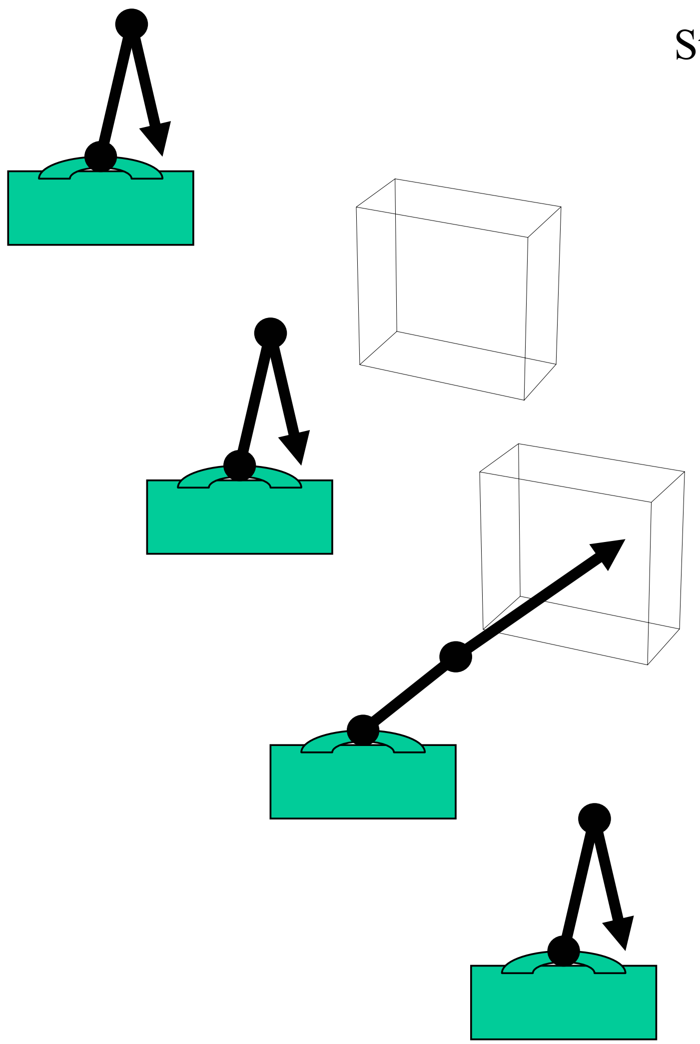
  1) Robustness is always an issue:
     - Modelling uncertainty.
     - Parameter variations.
     - Disturbances.
  2) Lack of theory (design uncertainty):
     - Relation between pole/zero locations and transient response.
     - Relation between Q/R weighting matrices in optimal control and transient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

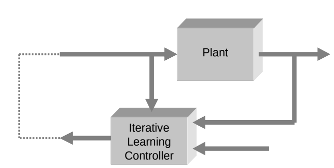


Step 1: Robot at rest, waiting for workpiece.

Step 2: Workpiece moved into position.

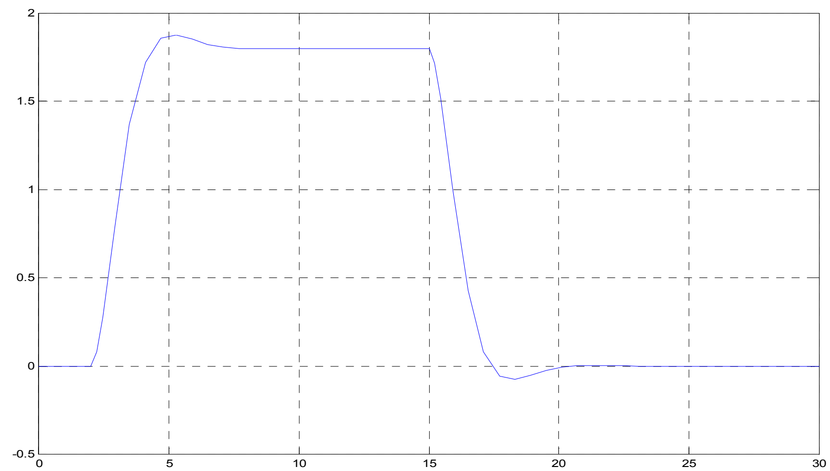
Step 3: Robot moves to desired location

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



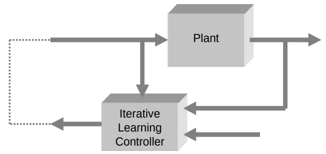
Errors are Repeated When Trajectories are Repeated

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



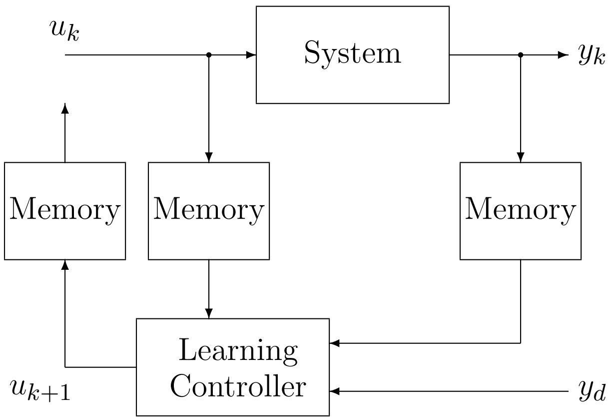
•Each time the system is operated it will see the same overshoot, rise time, settling time, and steady-state error.

•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.

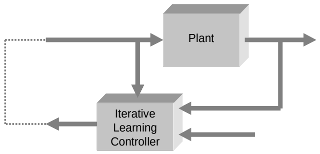


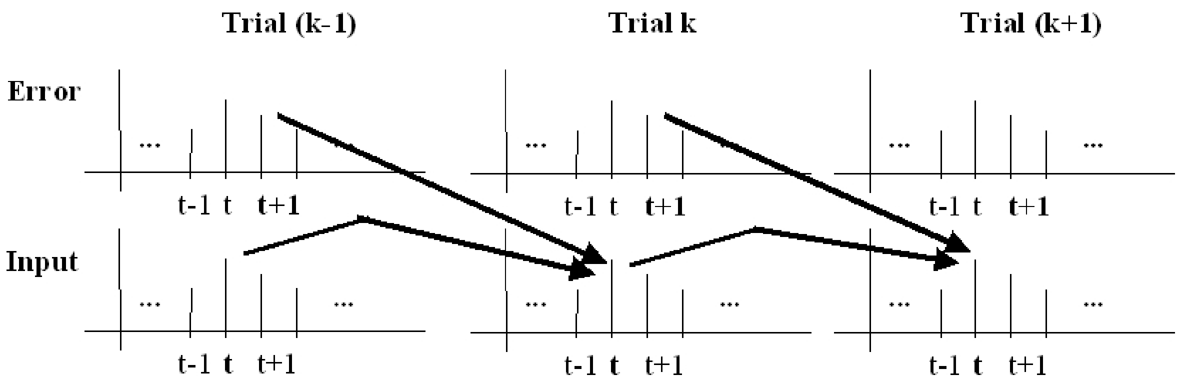
Iterative Learning Control

- Standard iterative learning control scheme:

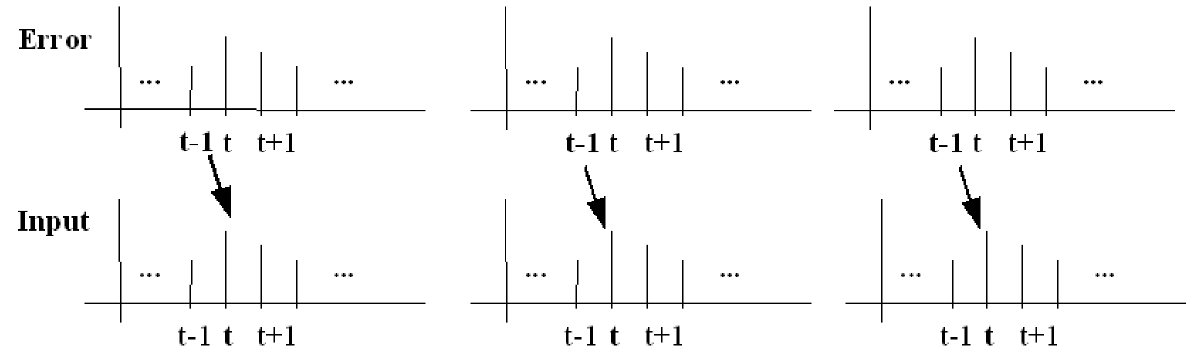


- A typical ILC algorithm has the form: $u_{k+1}(t) = u_k(t) + \gamma e_k(t+1)$.
- Standard ILC assumptions include:
  - Stable dynamics or some kind of Lipschitz condition.
  - System returns to the same initial conditions at the start of each trial.
  - Each trial has the same length.





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



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

A Simple Example

- For the nominal plant:

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

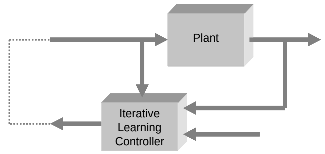
- Track the reference trajectory:

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

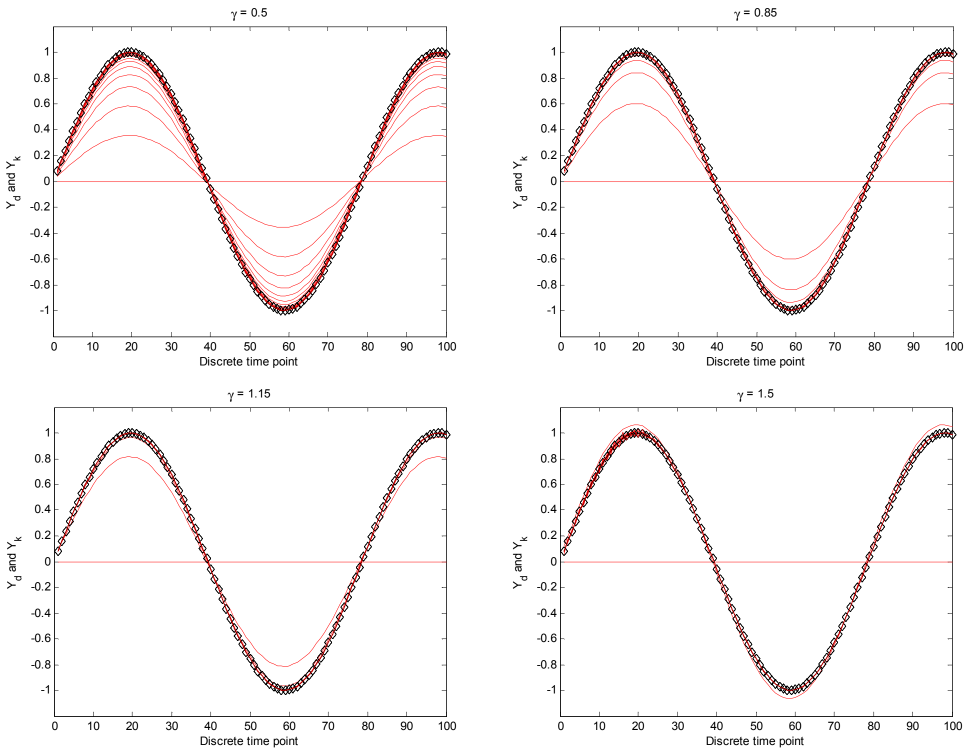
- We use the standard "Arimoto" algorithm:

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

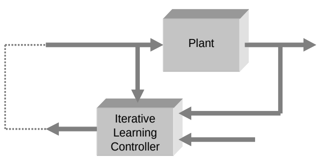
with four different gains: $\gamma = 0.5$, $\gamma = 0.85$, $\gamma = 1.15$, $\gamma = 1.5$



A Simple Example (cont.)

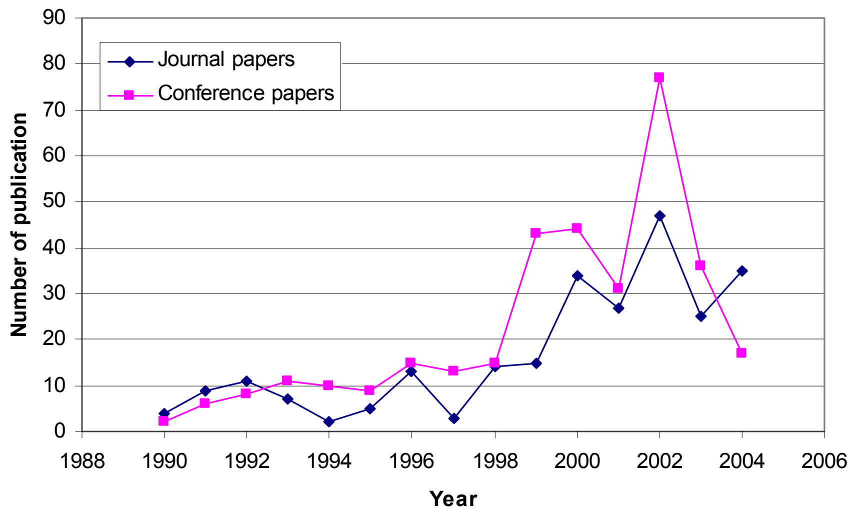


- In all cases, ILC converges well.
- Without knowing an accurate model of the plant, we achieve “perfect” tracking by iteratively updating the input from trial to trail.

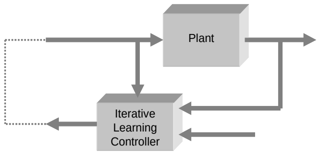


ILC Research History

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



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



First ILC paper- in Japanese (1978)

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．

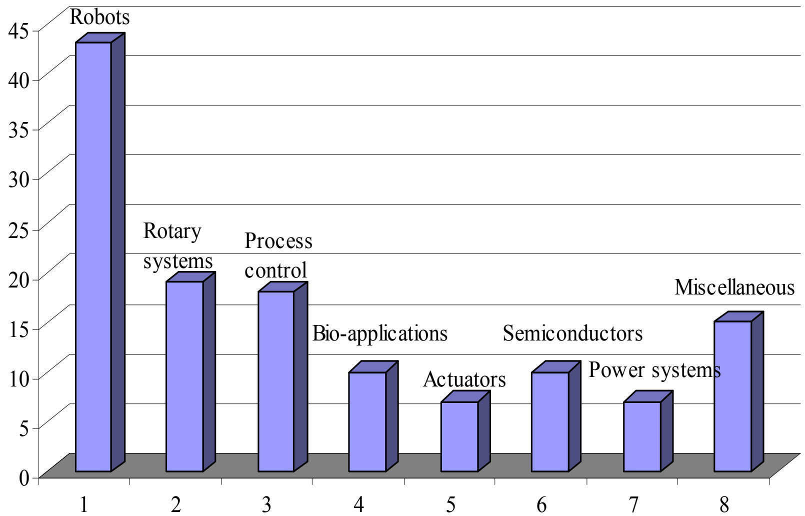
2.　修正アルゴリズム

2.1　問題設定

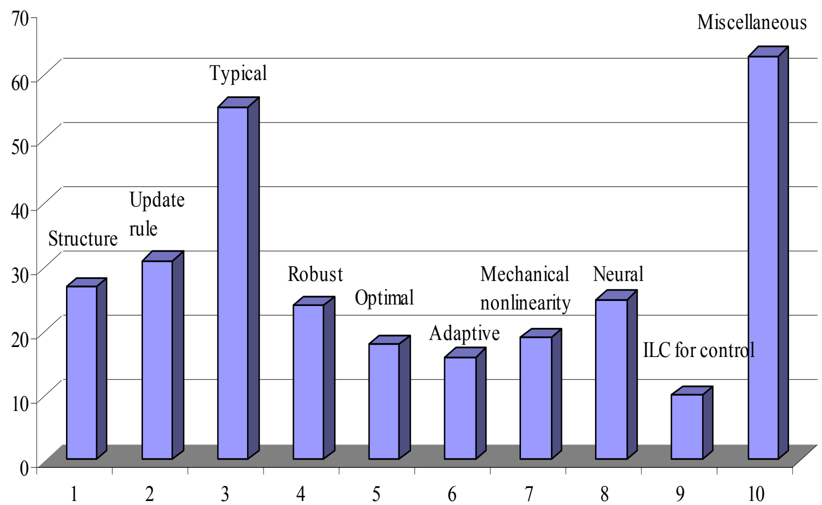
目標とする運動軌動を $\boldsymbol{\theta}_t(t)$ $(0 \leqq t \leqq T)$ としよう．ここで，$\boldsymbol{\theta}(t)$ は，人工の手の i 番目の関節角度を時間 t の関数として表した関数 $\theta_i(t)$ $(i=1\sim n)$ を第 i 成分とする n 次元ベクトルである． n は人工の手の自由度の数である．添字 t は $\boldsymbol{\theta}(t)$ が目標値関数であることを表す． T は運動時間である．試行においては，人工の手のサーボ系に適当な位置指令 $\boldsymbol{\theta}_{rj}(t)$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

ILC Research History (cont.)



By application areas.

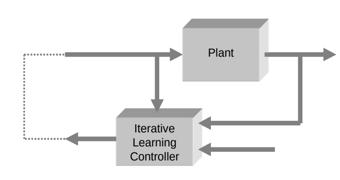


By theoretical areas.

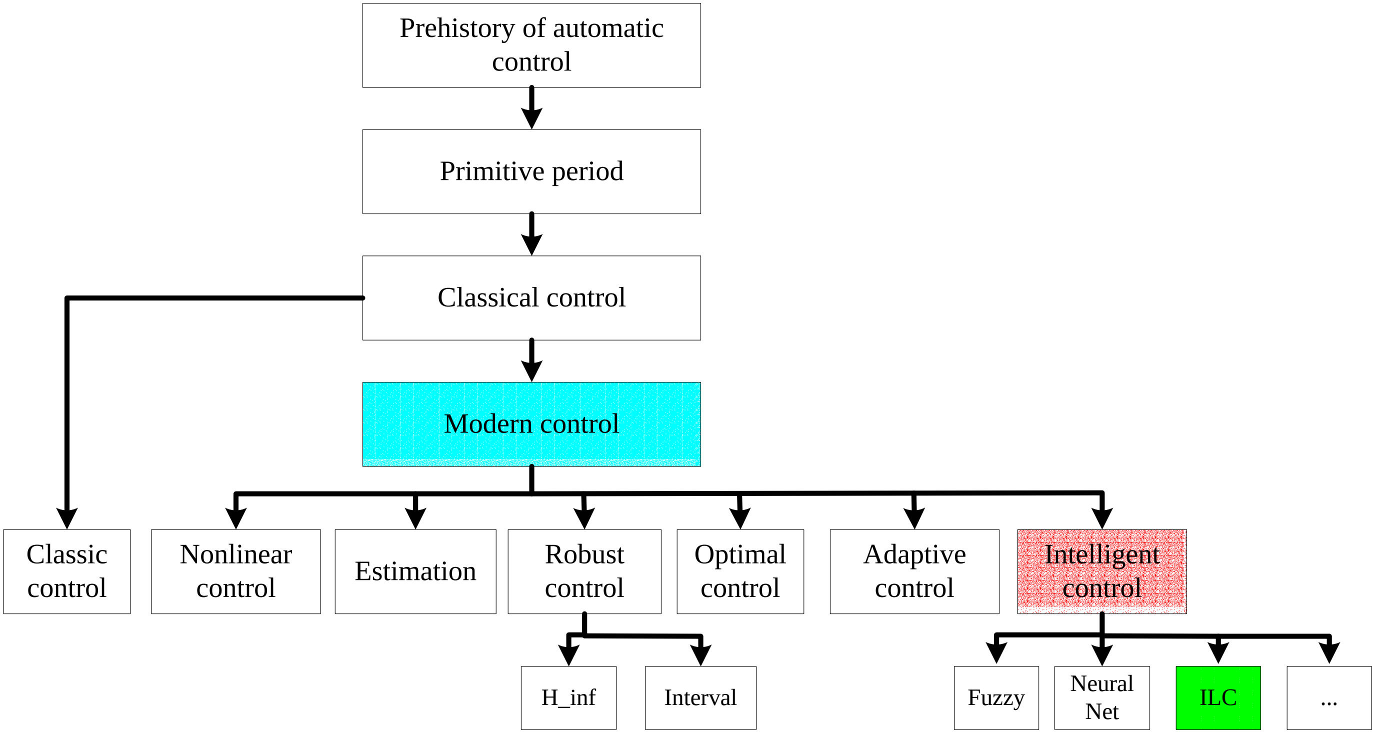
Selected ILC Industrial Applications

- 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 (described this afternoon).

See "Iterative Learning Control Survey: 1998-2004," Hyosung Ahn, CSOIS Technical Report # ..., Logan, UT 2005, for more information on the literature of ILC.



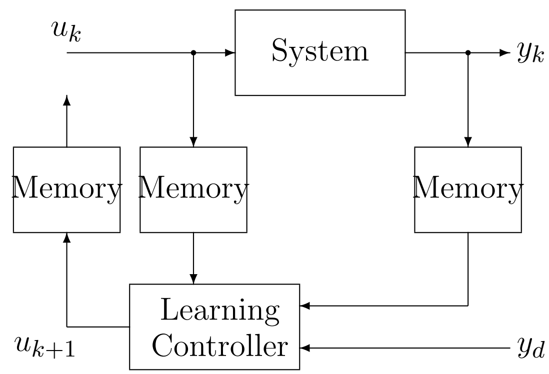
Control Engineering - History and ILC



**

ILC Problem Formulation

- Standard iterative learning control scheme:



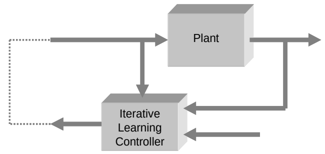
- **Goal**: Find a learning control algorithm

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

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

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

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



LTI ILC Convergence Conditions - 1

- **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:

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

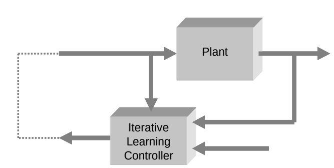
  It turns out that we can specify T_u and T_e in terms of T_n^* and the resulting learning controller converges to an optimal system input given by:

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

- **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.

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



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

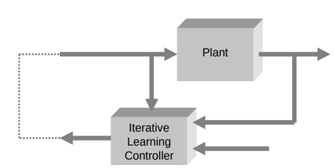
  - k denotes a trial (or execution, repetition, pass, etc.).
  - $t \in [0, N]$ denotes time (integer-valued).
  - $y_k(0) = y_d(0) = y_0$ for all k.

- 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.)

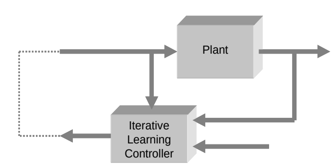
- Combine the plant equation with the ILC update rule to get:

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

- 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:

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

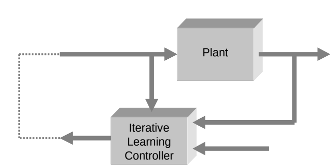
- By “lifting” along the time axis, for each trial k define:

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

- 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.)

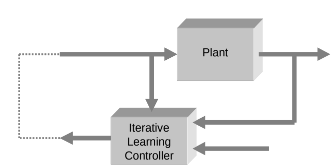
- For the linear, time-varying case, suppose we have the plant given by:

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

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:

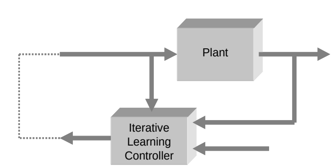
$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1)$$

  - 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:

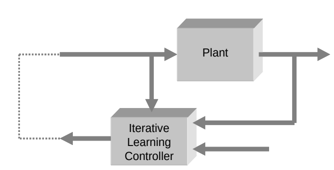
$$u_{k+1}(t) = u_k(t) + L(z)e_k(t+1)$$

  – 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:

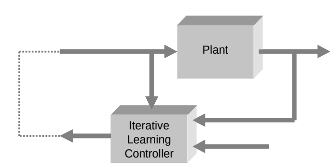
$$U_{k+1} = U_k + LE_k$$

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

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

- 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 ILC Design Problem

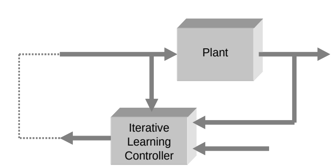
- The design of an ILC controller can be thought of as the selection of the matrix L.
  - For a causal ILC updating law, L will be in lower-triangular Toeplitz form.
  - For a noncausal ILC updating law, L will be in upper-triangular Toeplitz form.
  - For the popular zero-phase learning filter, L will be in a symmetrical band diagonal form.
  - L can also be fully populated.
- 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 $Q(z)$.



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.

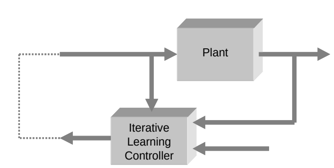
- Note that we can also write this as

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

where

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

.

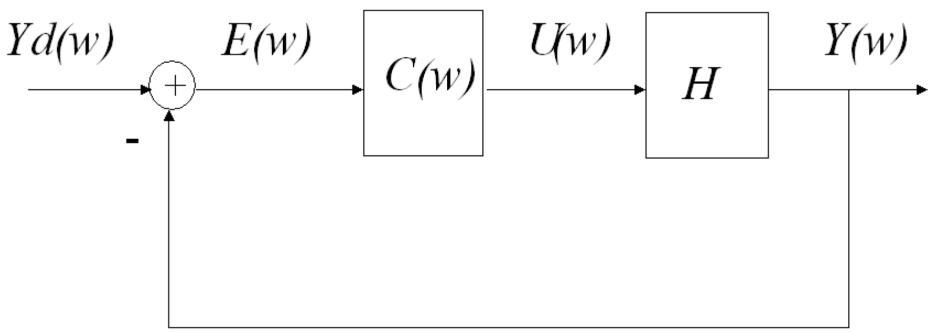


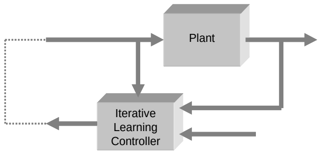
ILC as a MIMO Control System

- 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.

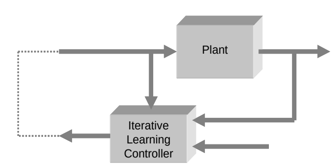
- For example, a "higher-order" ILC algorithm could have the form:

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

  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 $m = 1$):

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

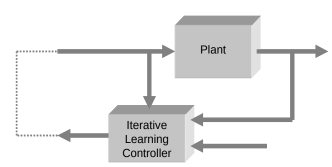
which has the supervector expression

$$\begin{aligned} U_{k+1} &= \bar{D}_nU_k + \bar{D}_{n-1}U_{k-1} + \cdots + \bar{D}_1U_{k-n+1} + \bar{D}_0U_{k-n} \\ &\quad + N_nE_k + N_{n-1}E_{k-1} + \cdots + N_1E_{k-n+1} + N_0E_{k-n} \end{aligned}$$

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

  - $U_{k+1} = U_k + LE_{k+1}$
  - $U_{k+1} = U_k + L_1E_k + L_0E_{k+1}$

  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:

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

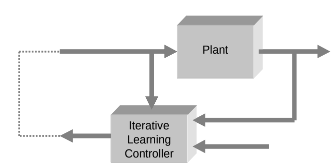
where

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

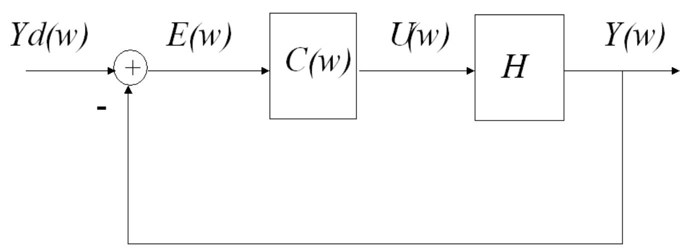
- This can be written in a matrix fraction as $U(w) = C(w)E(w)$ where:

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

- 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:

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

- Thus the ILC algorithm will converge (i.e., $E_k \to$ 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:

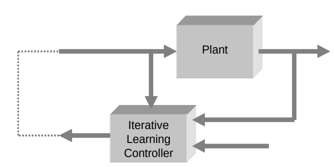
$$\begin{aligned} U_{k+1} &= (I - D_{n-1})U_k + (D_{n-1} - D_{n-2})U_{k-1} + \cdots \\ &\quad +(D_2 - D_1)U_{k-n+2} + (D_1 - D_0)U_{k-n+1} + D_0 U_{k-n} \\ &\quad +N_n E_k + N_{n-1}E_{k-1} + \cdots + N_1 E_{k-n+1} + N_0 E_{k-n} \end{aligned}$$

- 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:

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

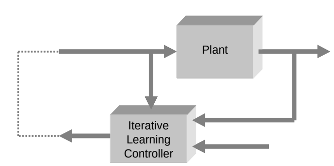
but where we now have:

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

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

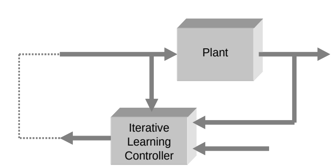
$$\begin{aligned} G_{cl}(w) &= H\left(I + \frac{I}{(w-1)}C(w)H\right)^{-1}\frac{I}{(w-1)}C(w), \\ &= H[(w-1)D_c(w) + N_c(w)H]^{-1}N_c(w) \end{aligned}$$

- 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 $E_k \rightarrow 0$ as long as the ILC algorithm converges (i.e., as long as G_{cl} is stable).



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

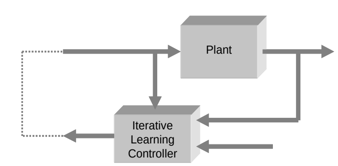
- In ILC, it is assumed that desired trajectory $y_d(t)$ and external disturbance are invariant with respect to iterations.
- When these assumptions are not valid, conventional integral-type, first-order ILC will no longer work well.
- In such a case, ILC schemes that are higher-order along the iteration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

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

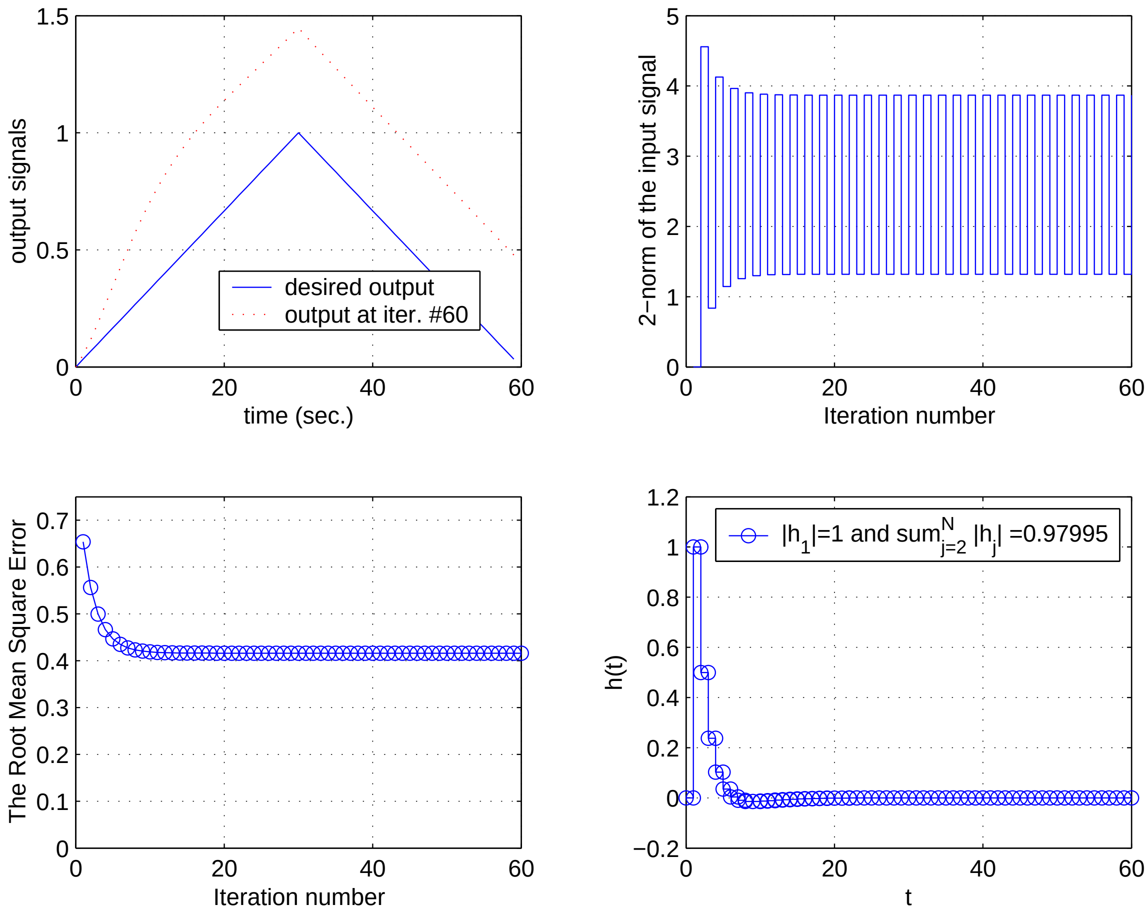
- Let the plant be subject to an additive output disturbance

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

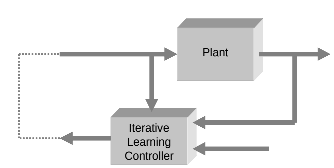
- This is an iteration-varying, alternating disturbance. If the iteration number k is odd, the disturbance is a positive constant in iteration k while when k is even, the disturbance jumps to a negative constant.
- In the simulation, we wish to track a ramp up and down on a finite interval.



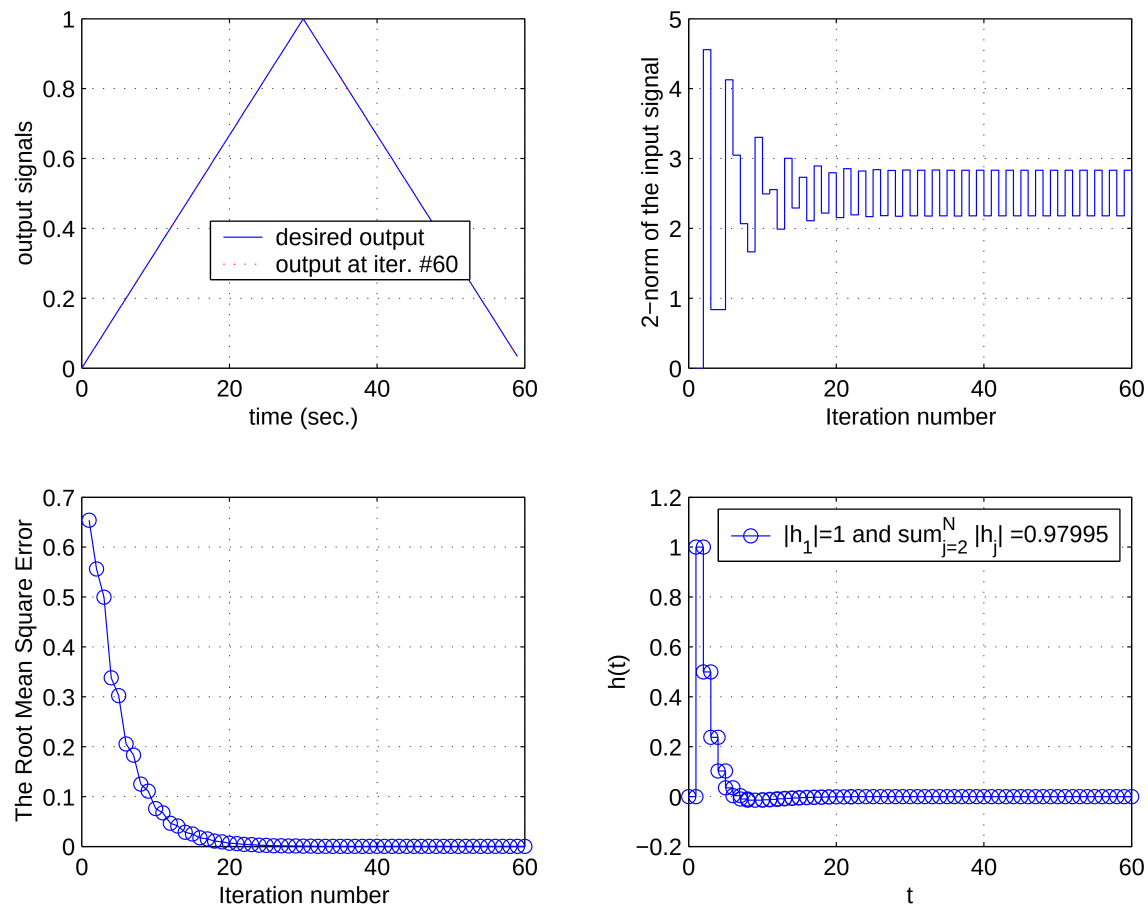
Example: First-Order ILC



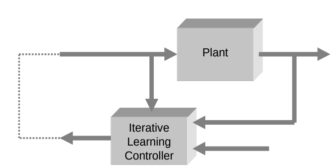
$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1), \gamma = 0.9 \Rightarrow C(w) = \frac{1}{(w-1)} L$$



Example: Second-Order, Internal Model ILC

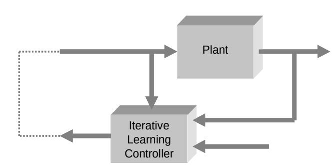


$$u_{k+1}(t) = u_{k-1}(t) + \gamma e_{k-1}(t+1) \text{ with } \gamma = 0.9 \Rightarrow C(w) = \frac{1}{(w^2-1)} L$$



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 the 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 the dynamic (in iteration) and uncertain MIMO system, such as

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

  or

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

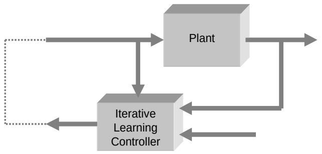
  or

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

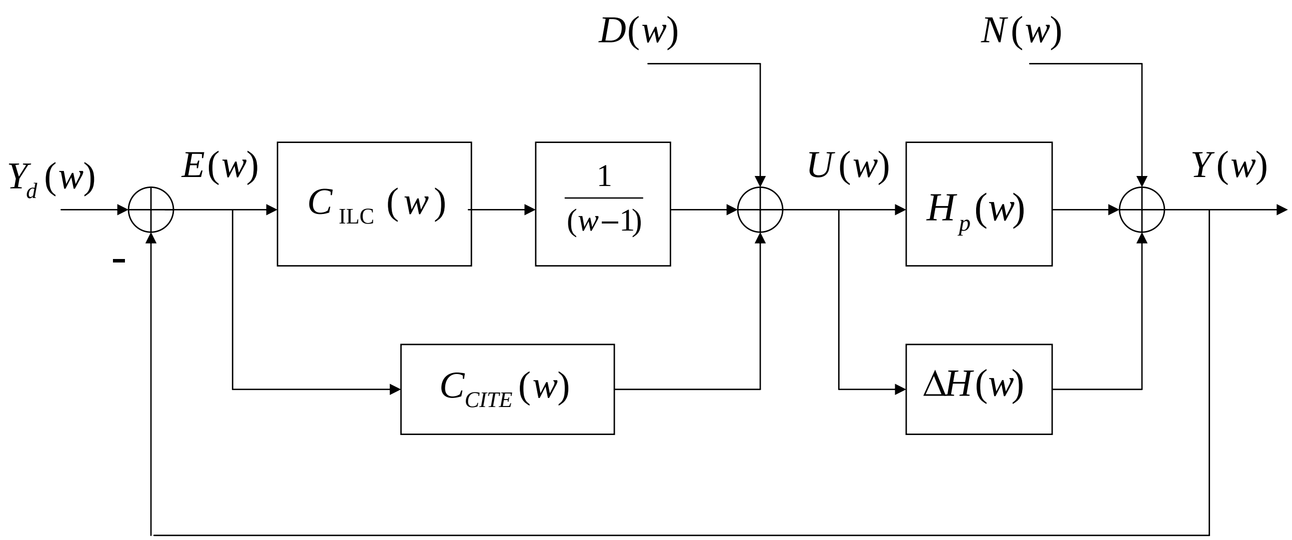
  or

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

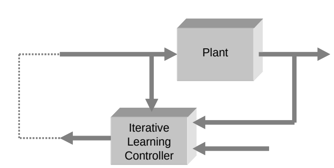
$$\cdots \text{ etc. } \cdots$$



Complete Framework

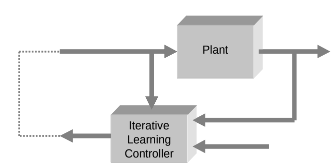


- $Y_d(w)$, $D(w)$ and $N(w)$ describe, respectively, the iteration-varying reference, disturbance, and noise signals. $H_p(w)$ denotes the (possibly iteration varying) plant.
- $\Delta H_p(w)$ represents the uncertainty in plant model, which may also be iteration-dependent.
- $C_{\text{ILC}}(w)$ denotes 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.



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

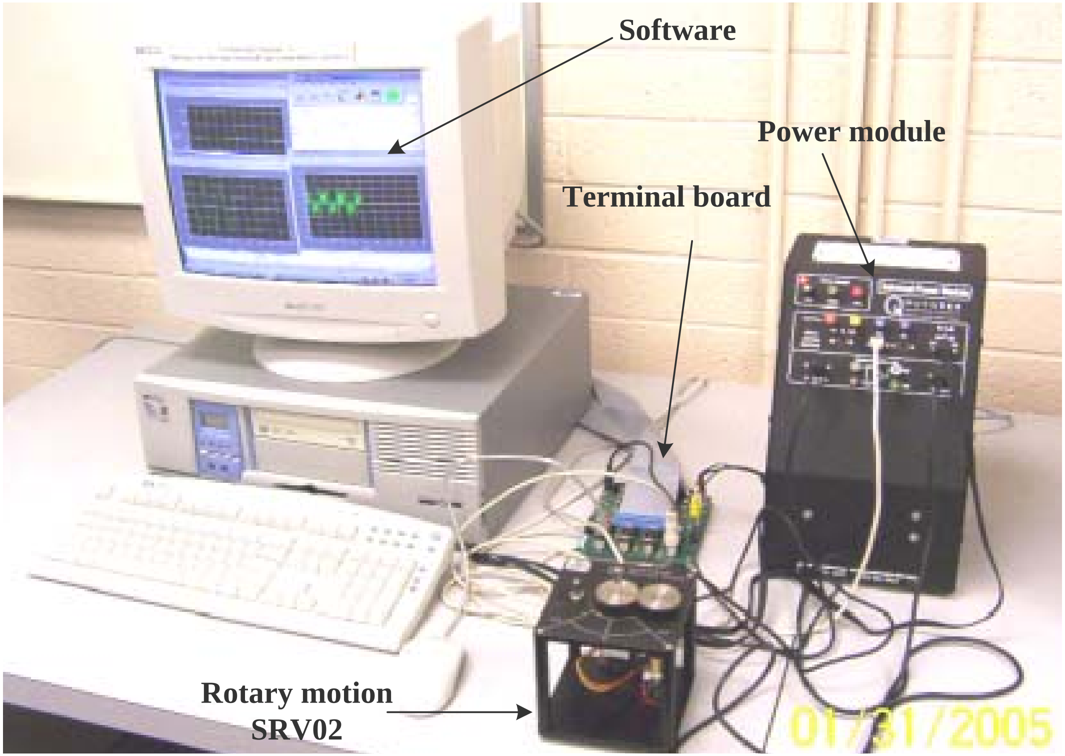


Categorization of algorithms

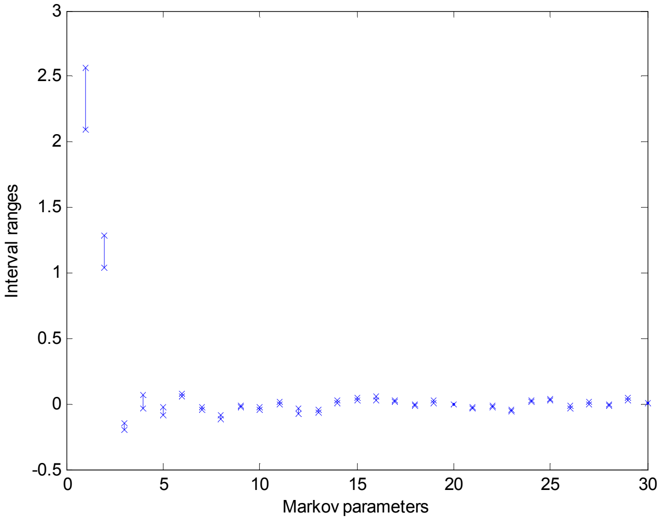
Y_r	D	N	C	H	
Constant	0	0	$\Gamma(w-1)^{-1}$	H_p	Classic approach (asymptotical)
Constant	0	0	$\Gamma(w-1)^{-1}$	H_p	Owens' multipass (asymptotical)
$Y_r(z)$	$D(w)$	0	$C(w)$	H_p	General (asymptotical)
$Y_r(z)$	$D(z)$	0	$C(w)$	$H(w)+\Delta(w)$	Frequency uncertainty. (asymptotical)
$Y_r(z)$	0	$w(t), v(t)$	$C(w)$	H_p	Stochastic ILC. (asym, monotone)
$Y_r(z)$	0	$w(t), v(t)$	$\Gamma(w-1)^{-1}$	H_p	Least quadratic ILC. (asym)
$Y_r(z)$	0	0	$\Gamma(w-1)^{-1}$	H^I	Interval (monotone)
$Y_r(z)$	$D(z)$	$w(t), v(t)$	$C(w)$	$H(z)+\Delta H(z)$	Time domain H_∞ (asymptotical)
$Y_r(z)$	$D(w)$	$w(k,t), v(k,t)$	$C(w)$	$H(w)+\Delta H(w)$	Iterative domain H_∞ (asymptotical)
$Y_r(z)$	0	$w(k,t), v(k,t)$	$\Gamma(k)$	$H_p+\Delta H(k)$	Iteration varying uncertainty (monotone)
$Y_r(z)$	0	$\widetilde{H}$	Γ	H_p	Intermittent measurement
$Y_r(w)$	0	0	$\Gamma(w-1)^{-1}$	H_p	Periodically-I.V.-reference(asymptotical)
$Y_r(w)$	0	0	$\Gamma(w-1)^{-1}$	H_p	Periodically-I.V.-reference(monotonic)

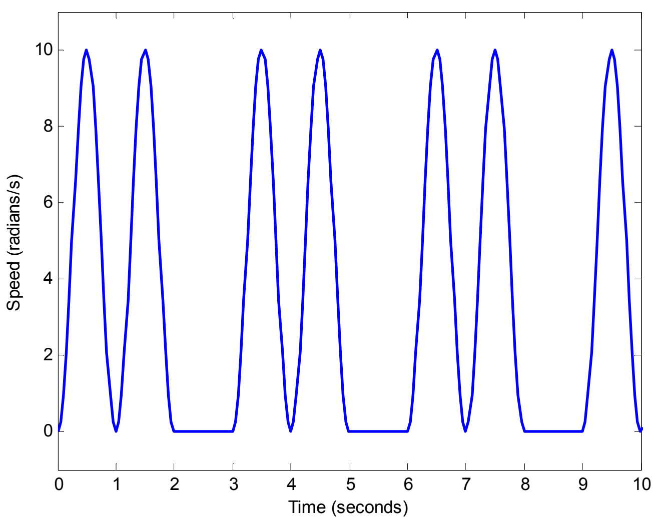
**

Experimental test: Setup



Impulse responses and desired trajectory





Achieved results