Iterative Learning Control (ILC) [H04Q7]

Jan Swevers

July 2008

Contents of chapter

This lecture is based on ”A survey of iterative learning control (a learning-based method for high-performance tracking control)”, by D. A. Bristow, M. Tharayil, and A. Alleyne, IEEE Control Systems Magazine, June 2006.

Introduction

- Based on the notion that the performance of a system that executes the same task multiple times can be improved by learning form the previous executions (iterations).
- Examples: basketball player shooting a free throw from a fixed position, industrial robot that has to repeat the same task over and over again.
- We consider learning controllers for systems that perform the same operation repeatedly and under the same operating conditions.
- In that case, a **non**learning controller yields the same tracking error on each pass, that is, the error signals from previous iterations are not used ...

Objective of ILC

- To improve performance by incorporating error information into the control for subsequent iterations.
- Achieve high performance with low transient tracking error despite **large model uncertainty** and **repeating disturbances**.

Relation with repetitive control (RC)

- RC is intended for continuous operation, whereas ILC is intended for discontinuous operation: the difference is the setting of the initial conditions for each trial [redacted]
- E.g. ILC: to control a robot that performs a task and returns to its home position, and comes to rest before repeating the task again; the initial conditions are set to the same value on each trial.
- E.g. RC: to control a hard disk drive's read/write head, in which each iteration is a full rotation of the disk, and the next iteration follows the current iteration; the initial conditions are set to the final conditions of the previous trial.

Applications of ILC

- Industrial applications where mass production entails repetition.
- Examples: industrial robotics, computer numerical control (CNC) machine tools, wafer stage motion systems, injection-molding machines, aluminum extruders, camless engine valves, thermal processing, semibatch chemical reactors.
- ILC is also applied to systems that do not have identical repetition, but where the different tasks can be equalized by a time-scale transformation [redacted]
- Basic [redacted]8 journal article written in Japanese.

Goal of this ILC lecture

- Provide a tutorial that gives a complete picture of the benefits, limitations.
- Intended for the practicing engineer: learn to design and analyze a simple ILC
- We restrict ourselves to SISO discrete-time systems: notation and formulations are more accessible.

ILC versus good feedback and feedforward design

- Feedback control:
  - A feedback controller reacts to inputs and disturbances and, therefore, always has a lag in transient tracking.
  - A feedback controller can accommodate variations or uncertainties in the system model.
  - Robustness and stability problems can occur: system model/dynamics have to be known to some extend
  - "Good" performance at the first trial, but no improvement after that.

- Feedforward
  - Feedforward control can eliminate this lag, but only for known or measurable signals, such as the reference, and typically not for disturbances.
  - A feedforward controller performs well only to the extent that the system is accurately known.

- ILC:
  - Generates a feedforward control that tracks a specific reference or rejects a repeating disturbance.
  - Is anticipatory and can compensate for exogenous signals, such as repeating disturbances, in advance by learning from previous iterations.
  - Does not require that the exogenous signals (references or disturbances) be known or measured.
  - ILC learns feedforward through practice and it is highly robust to system uncertainties.

ILC overview

- ILC system description
- General ILC algorithms

ILC system description

- Consider the folowing discrete-time linear time-invariant (LTI) SISO system:

$$y_j(k) = P(q)u_j(k) + d(k) \tag{1}$$

[redacted]

  where k is the time index, j is the iteration index, q is the forward time-shift operator: $qx(k) = x(k+1)$ or $q^{-1}x(k) = x(k-1)$.

- y_j is the output, u_j is de control input, and d is an exogenous signal that repeats each iteration.

- $P(q)$ is the system with a relative degree of m, is asymptotically stable (when $P(q)$ is not stable, it has to be stabilized first).

[redacted]

- A system delay $m = 1$ sample is assumed in most of the cases.

ILC system description (2)

- Next consider the N-sample sequence of inputs, outputs, and desired output

[redacted]

$$\begin{array}{rcl} u_j(k) & , & k \in \{0,\, 1,\, \ldots,\, N-1\} \\ {}_j(k) & , & k \in \{m,\, m+1,\, \ldots,\, N+m-1\} \\ d(k) & , & k \in \{m,\, m+1,\, \ldots,\, N+m-1\} \\ y_d(k) & , & k \in \{m,\, m+1,\, \ldots,\, N+m-1\} \end{array}$$

- The performance or error signal is defined by $e_j(k) = y_d(k) - y_j(k)$.

ILC system description (3)

- $P(q)$ can be in transfer function or state space form.
- To cope with non-zero initial conditions, it is easier to use the state space form:

$$
\begin{aligned}
x_j(k+1) &= Ax_j(k) + Bu_j(k) \qquad (2)\\
y_j(k) &= Cx_j(k) \qquad (3)
\end{aligned}
$$

with $x_j(0) = x_0$ for all j. This state-space system is equivalent to:

$$y_j(k) = \underbrace{C(qI - A)^{-1}B}_{P(q)}\, u_j(k) + \underbrace{CA^k x_0}_{d(k)},$$ [redacted]

that is, the repeating nonzero initial condition is captured in $d(k)$.

ILC system description (4)

- A widely used ILC learning algorithm is: [redacted]

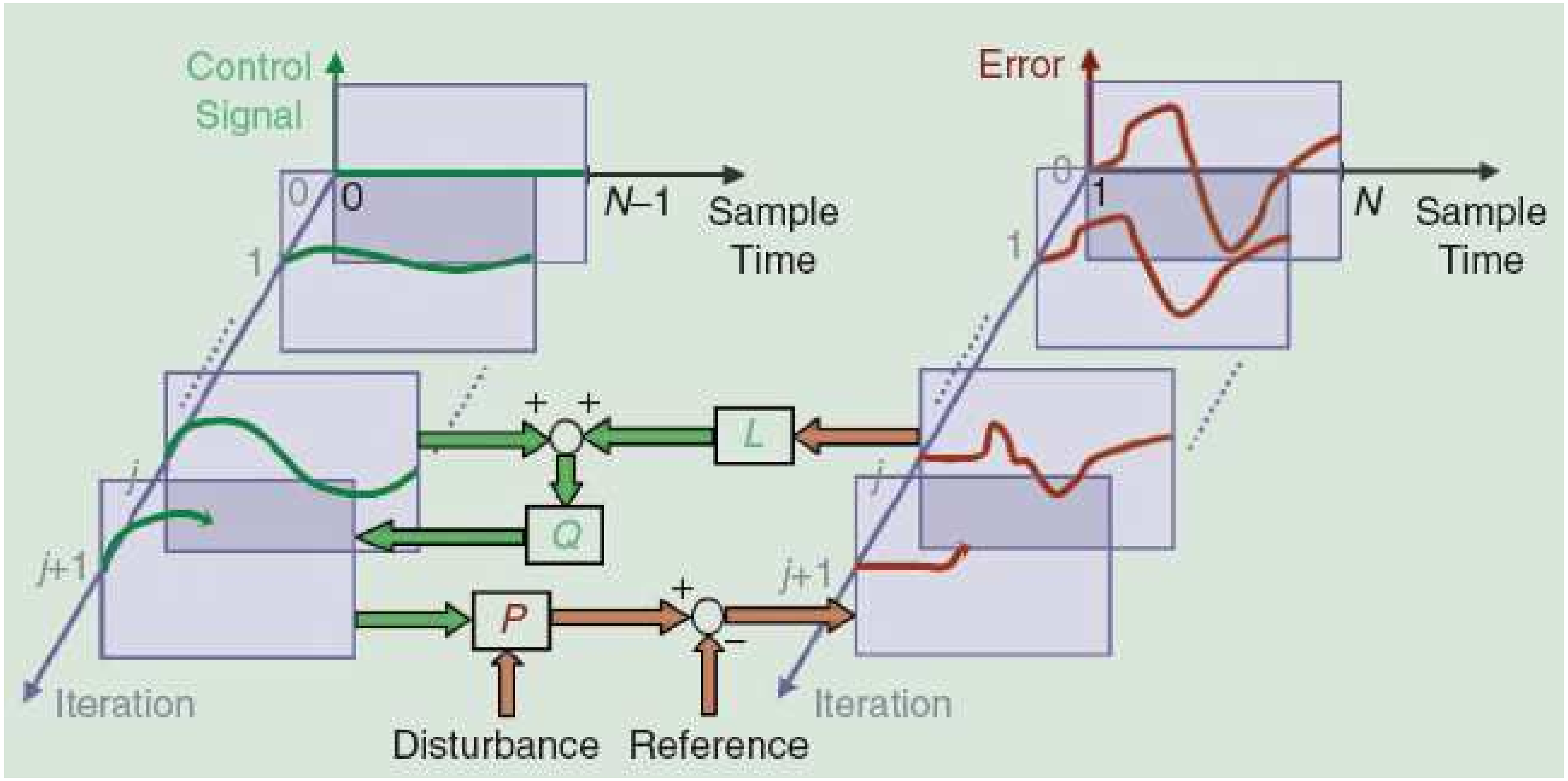
$$u_{j+1}(k) = Q(q)\left[u_j(k) + L(q)e_j(k+1)\right], \tag{4}$$

[redacted] [redacted]

  where the LTI filters $Q(q)$ and $L(q)$ are defined as Q-filter and learning filter, respectively.

- The following slide illustrates this two-dimensional (2-D) ILC system.

ILC system description (5)



Schematic representation of a two-dimensional, first-order ILC system

General ILC algorithms

- There are several possible variations to the ILC algorithm (4).
  - linear time varying (LTV) functions,
  - nonlinear functions,
  - iteration-varying functions,
  - high-order learning algorithms that use u_i and e_i for $i \in \{j - N_0 + 1, \ldots, j\}$ and $N_0 > 1$ to calculate u_{j+1}.
  - the current error $e_{j+1}(k)$ can also be used to calculate $u_{j+1}(k)$ to obtain a current-iteration learning algorithm. This is equivalent to ILC algorithm (4) combined with a feedback controller on the system.

System representations

Analysis of ILC systems is based on two system representations

- Time-domain analysis using the lifted-system framework
- Frequency-domain analysis using the z-domain representation

Time-domain analysis using the lifted-system framework

- The lifted-system framework is based on the impulse response representation of the system:

$$P(q) = p_1 q^{-1} + p_2 q^{-2} + p_3 q^{-3} + \dots\,, \tag{5}$$

  where the coefficients p_k are the Markov parameters. Note that $p_1 \neq 0$ since $m = 1$.

- For the state space description (2) and (3), $p_k = CA^{k-1}B$.

Time-domain analysis using the lifted-system framework (2)

- Stacking the signals in column vectors, the system dynamics (1) can be written as the following $N \times N$-dimensional lifted system:

$$\underbrace{\begin{bmatrix} y_j(1) \\ y_j(2) \\ \vdots \\ y_j(N) \end{bmatrix}}_{\mathbf{y}_j} = \underbrace{\begin{bmatrix} p_1 & 0 & \dots & 0 \\ p_2 & p_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_N & p_{N-1} & \dots & p_1 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} u_j(0) \\ u_j(1) \\ \vdots \\ u_j(N-1) \end{bmatrix}}$$

[redacted]

$$+ \underbrace{\begin{bmatrix} d(1) \\ d(2) \\ \vdots \\ d(N) \end{bmatrix}}_{\mathbf{d}}, \qquad (6)$$

Time-domain analysis using the lifted-system framework (3)

- tracking error

$$\underbrace{\begin{bmatrix} e_j(1) \\ e_j(2) \\ \vdots \\ e_j(N) \end{bmatrix}}_{\mathbf{e}_j} = \underbrace{\begin{bmatrix} y_d(1) \\ y_d(2) \\ \vdots \\ y_d(N) \end{bmatrix}}_{\mathbf{y}_d} - \underbrace{\begin{bmatrix} y_j(1) \\ y_j(2) \\ \vdots \\ y_j(N) \end{bmatrix}}_{\mathbf{y}_j}.$$

- Remark that if the time delay is $m > 1$, the vectors $\mathbf{e}_j$, $\mathbf{y}_d$, $\mathbf{y}_j$, and $\mathbf{d}$, change, that is, their first element is the variable with time index $k = m$ instead of $k = 1$. Also the matrix $\mathbf{P}$ changes: just replace all coefficients p_i by p_{m+i-1}.

Time-domain analysis using the lifted-system framework (4)

- Likewise, the learning algorithm (4) can be written in this lifted form. The Q-filter and learning filter can be noncausal functions with the following impulse responses: [redacted]

  $$Q(q) = \ldots + q_{-2}q^{2} + q_{-1}q^{1} + q_0 + q_1 q^{-1} + q_2 q^{-2} + \ldots$$

  and

  $$L(q) = \ldots + l_{-2}q^{2} + l_{-1}q^{1} + l_0 + l_1 q^{-1} + l_2 q^{-2} + \ldots .$$

Time-domain analysis using the lifted-system framework (5)

- In lifted form, equation (4) becomes:

$$\underbrace{\begin{bmatrix} u_{j+1}(0) \\ \vdots \\ u_{j+1}(N-1) \end{bmatrix}}_{\mathbf{u}_{j+1}} = \underbrace{\begin{bmatrix} q_0 & q_{-1} & \cdots & q_{-N+1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N-1} & q_{N-2} & \cdots & q_0 \end{bmatrix}}_{\mathbf{Q}} \left(\underbrace{\begin{bmatrix} u_j(0) \\ \vdots \\ u_j(N-1) \end{bmatrix}}_{\mathbf{u}_j} \right.$$

$$\left. + \underbrace{\begin{bmatrix} l_0 & l_{-1} & \cdots & l_{-N+1} \\ \vdots & \vdots & \ddots & \vdots \\ l_{N-1} & l_{N-2} & \cdots & l_0 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} e_j(1) \\ \vdots \\ e_j(N) \end{bmatrix}}_{\mathbf{e}_j} \right). \qquad (7)$$

Time-domain analysis using the lifted-system framework (6)

- When $Q(q)$ and $L(q)$ are causal filters, it follows that $q_{-1} = q_{-2} = \ldots = 0$ and $l_{-1} = l_{-2} = \ldots = 0$, and the matrices $\mathbf{Q}$ and $\mathbf{L}$ are lower triangular.
- The matrices $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{L}$ are Toeplitz, meaning that all of the entries along each diagonal are identical.
- This lifted framework can very easily accomodate an LTV plant, Q-filter, and learning filter: this yields similar expressions but the matrices $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{L}$ will not be Toeplitz anymore.

Frequency-domain analysis using the z-domain representation

- To apply the z-transformation to the ILC system (1) and (4), we must have $N = \infty$ because the z-transform requires that the signals be defined over an infinite time horizon.
- Since all practical applications of ILC have a finite horizon, the z-domain representation is an approximation of the ILC system.
- The z-transform of (1) and (4) are:

$$\begin{aligned} Y_j(z) &= P(z)U_j(z) + D(z), && (8) \\ U_{j+1}(z) &= Q(z)\left[U_j(z) + zL(z)E_j(z)\right], && (9) \end{aligned}$$

where $E_j(z) = Y_d(z) - Y_j(z)$. The z that multiplies $L(z)$ emphasizes the forward time shift used in learning!

Analysis

- Stability
- Performance [redacted]
- Transient learning behavior [redacted]
- Robustness [redacted]

Stability

- The ILC system (1) and (4) is *asymptotically stable* (AS) if there exists $\bar{u} \in \Re$ such that

$$|u_j(k)| \leq \bar{u}, \text{ for all } k = \{0, \ldots, N-1\} \text{ and } j = \{0, 1, \ldots, \},$$

and, for all $k \in \{0, \ldots, N-1\}$,

$$\lim_{j \to \infty} u_j(k) \text{ exists.}$$

- We define the converged control as

$$u_\infty(k) = \lim_{j \to \infty} u_j(k).$$

- Time-domain and frequency-domain conditions for AS will be discussed in the following slides

Stability (2)

- Substituting $\mathbf{e}_j = \mathbf{y}_d - \mathbf{y}_j$ into the system dynamics (6) and into the learning algorithm (7) yields the following closed-loop iteration domain dynamics:

$$\mathbf{u}_{j+1} = \mathbf{Q}(\mathbf{I} - \mathbf{L}\mathbf{P})\mathbf{u}_j + \mathbf{Q}\mathbf{L}(\mathbf{y}_d - \mathbf{d}) \tag{10}$$

- Let $\rho(\mathbf{A}) = \max_i |\lambda_i(A)|$ be the spectral radius of the matrix $\mathbf{A}$ and $\lambda_i(\mathbf{A})$ the ith eigenvalue of $\mathbf{A}$.

Stability (3)

- **THEOREM 1**

  The ILC system (1) and (4) is AS if and only if

$$\rho(\mathbf{Q}(\mathbf{I} - \mathbf{LP})) < 1. \tag{11}$$

- When the Q-filter and the learning filter are causal, the matrix $\mathbf{Q}(\mathbf{I} - \mathbf{LP})$ is lower triangular and Toeplitz with repeated eigenvalues

$$\lambda = q_0(1 - l_0 p_1). \tag{12}$$

  In this case, (11) is equivalent to the scalar condition:

$$|q_0(1 - l_0 p_1)| < 1. \tag{13}$$

Stability (4)

- Based on (8) and (9), the iteration domain dynamics for the z-domain representation are:

$$U_{j+1}(z) = Q(z)[1 - zL(z)P(z)]U_j(z) + zQ(z)L(z)[Y_d(z) - D(z)]. \tag{14}$$

- **THEOREM 2**
  If
  $$\|Q(z)[1 - zL(z)P(z)]\|_\infty < 1, \tag{15}$$
  then the ILC system with $N = \infty$ is AS.

- If $Q(z)$ and $L(z)$ are causal filters, (15) also implies AS for finite-duration ILC systems.

- This stability condition (15) is only sufficient and in general much more conservative than the necessary and sufficient condition (11).

Performance

- The performance of the ILC system is measured by looking at the asymptotic value of the error:

$$\begin{aligned} e_\infty(k) &= \lim_{j\to\infty} e_j(k) \\ &= \lim_{j\to\infty} (y_d(k) - P(q)u_j(k) - d(k)) \\ &= y_d(k) - d(k) - P(q)u_\infty(k). \end{aligned}$$

- If the ILC system is AS, the asymptotic error is

$$\mathbf{e}_\infty = [\mathbf{I} - \mathbf{P}[\mathbf{I} - \mathbf{Q}(\mathbf{I} - \mathbf{LP})]^{-1}\mathbf{QL}](\mathbf{y}_d - \mathbf{d}) \qquad (16)$$

for the lifted system and

[redacted]

$$E_\infty = \frac{1 - Q(z)}{1 - Q(z)[1 - zL(z)P(z)]}[Y_d(z) - D(z)] \qquad (17)$$

for the z-domain representation.

Performance (2)

- **THEOREM 3**
  Suppose that P and L are not identically zero. Then, $e_\infty = 0$ for all k and for all y_d and d, if and only if the system is AS and $Q(q) = 1$.
- Many ILC algorithms set $Q(q) = 1$ and thus do not include Q-filtering, and consequently yield perfect performance. However, Q-filtering can improve transient learning behavior and robustness ... see later.
- In most applications, the Q-filter is a low-pass filter with DC-gain equal to 1. As a result, it yields perfect performance at low frequencies and gradually (as frequency increases) shuts off the ILC algorithm to achieve a tracking error equal to $Y_d(e^{j\omega}) - D(e^{j\omega})$ at high (infinite) frequencies ω.

Transient learning behavior

Example

- Consider the following system and ILC algorithm

$$\begin{aligned} y(k) &= \frac{q}{(q-0.9)^2}u(k) \\ u_{j+1}(k) &= u_j(k) + 0.5e_j(k+1) \end{aligned}$$

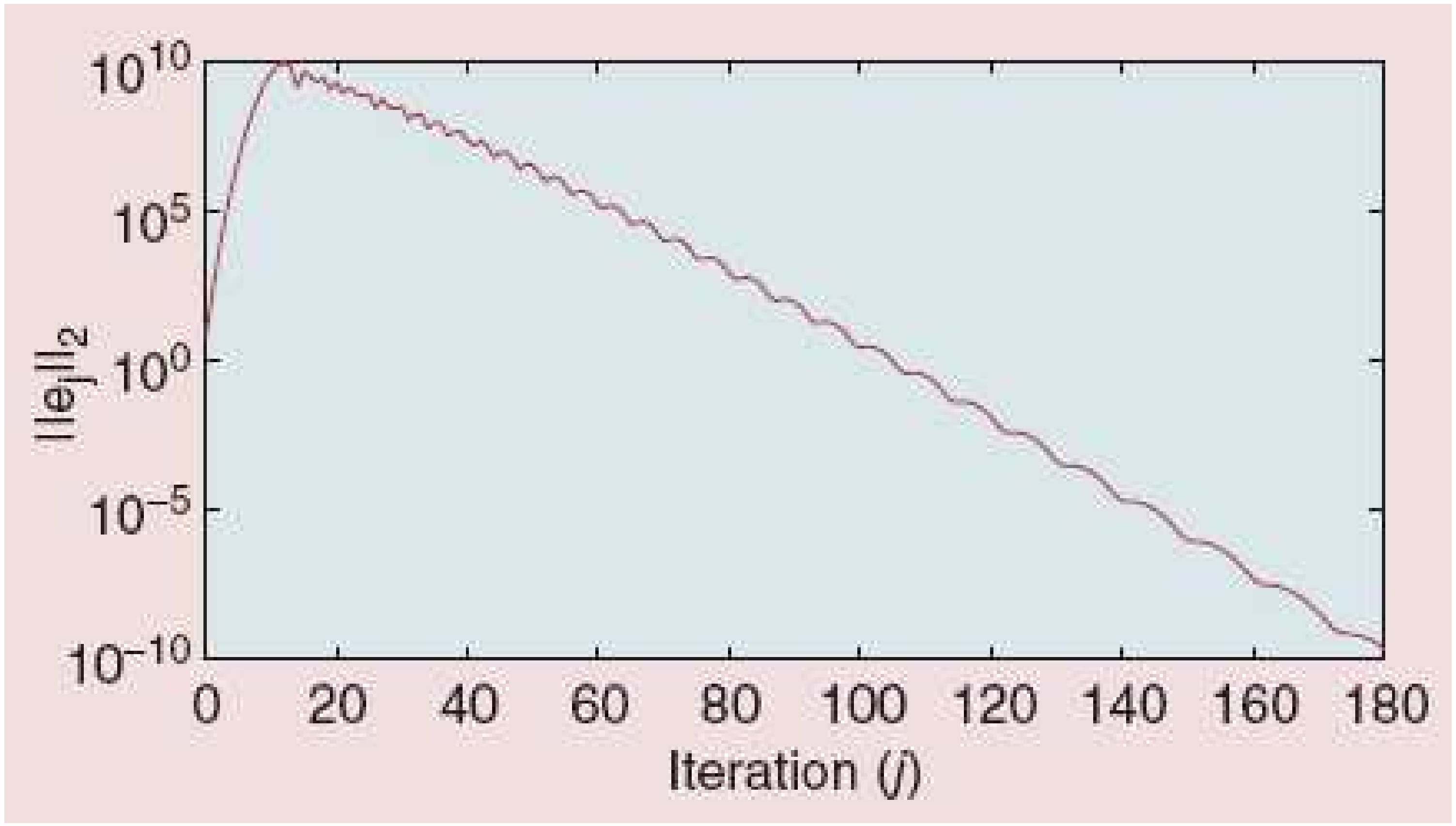
- $Q(q) = 1$ and $L(q) = 0.5$, both are causal, and $p_1 = 1$.

[redacted]

- All the eigenvalues of the lifted system (12) are equal to $\lambda = 0.5$, that is, the ILC system is AS (theorem 1).
- $N = 50$. The asymptotic error is 0, because $Q(q) = 1$.
- y_d is a smooth trajectory, the tracking error increases over the first 12 iterations by nine orders of magnitude: very bad transient behavior.

[redacted]

Transient learning behavior (2)



Transient learning behavior (3)

- Transient behavior can be bad, even if the stability condition(s) is(are) satisfied
- It is difficult to distinguish transient growth from instability in practice.
- Large transient growth is a fundamental topic of ILC and preventing it is an essential objective in ILC design.
- To avoid large learning transients, *monotonic* convergence is desirable.

Transient learning behavior (4)

- The system (1) and (4) is monotonically convergent under a given norm $\| \bullet \|$ if
$$\|e_\infty - e_{j+1}\| \leq \gamma \|e_\infty - e_j\|,$$
for all $j \in \{1, 2, \ldots, \}$, where $0 \leq \gamma < 1$ is the convergence rate.

- In the lifted-system representation:
$$\mathbf{e}_\infty - \mathbf{e}_{j+1} = \mathbf{PQ}(\mathbf{I} - \mathbf{LP})\mathbf{P}^{-1}(\mathbf{e}_\infty - \mathbf{e}_j). \tag{18}$$

- When $P(q)$, $Q(Q)$ and $L(q)$ are causal, the matrices $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{L}$ commute and (18) reduces to:
$$\mathbf{e}_\infty - \mathbf{e}_{j+1} = \mathbf{Q}(\mathbf{I} - \mathbf{LP})(\mathbf{e}_\infty - \mathbf{e}_j). \tag{19}$$

Transient learning behavior (5)

- For the z-domain representation, the error dynamics equal:

$$E_\infty(z) - E_{j+1}(z) = Q(z)[1 - zL(z)P(z)][E_\infty(z) - E_j(z)]. \tag{20}$$

- Let $\bar{\sigma}(.)$ be the maximum singular value and let us consider the 2-norm to measure convergence, then the following monotonic convergence conditions can be derived.

- **THEOREM 4**
  If the ILC system (1) and (4) satisfies

$$\gamma_1 \triangleq \bar{\sigma}\left(\mathbf{PQ}(\mathbf{I} - \mathbf{LP})\mathbf{P}^{-1}\right) < 1, \tag{21}$$

  then

$$\|e_\infty - e_{j+1}\|_2 \leq \gamma_1 \|e_\infty - e_j\|_2$$

  for all $j \in \{1, 2, \ldots, \}$.

Transient learning behavior (6)

- **THEOREM 5**

  If the ILC system (1) and (4) with $N = \infty$ satisfies

$$\gamma_2 \triangleq \|Q(z)[1 - zL(z)P(z)]\|_\infty < 1, \tag{22}$$

  then

$$\|E_\infty(z) - E_{j+1}(z)\|_\infty < \gamma_2 \|E_\infty(z) - E_j(z)\|_\infty$$

  for all $j \in \{1, 2, \ldots, \}$.

- When $Q(z)$ and $L(z)$ are causal filters, (22) also implies that

$$\|e_\infty - e_{j+1}\|_2 \leq \gamma_2 \|e_\infty - e_j\|_2$$

  for all $j \in \{1, 2, \ldots, \}$ for the ILC system with a finite duration N.

Transient learning behavior (7)

- Note that the z-domain monotonic convergence condition (22) is identical to the stability condition (15) (theorem 2). Thus, when $Q(z)$ and $L(z)$ are causal filters, the stability condition (15) provides stability and monotonic convergence independent of N.
- The monotonic convergence condition of the lifted system (21) is a more stringent requirement than the stability condition (11), and both are specific to the iteration duration N under consideration.

Robustness

- Robustness is an important issue for ILC, because if the system model would be known exactly other, more direct approaches for "perfect" tracking are preferable.
- Here we consider robustness as related to stability and monotonic convergence.
- **Example**: consider $Q(q) = 1$ (zero converged error), and a causal $L(q)$.
  - The AS condition is $|1 - l_0 p_1| < 1$, which (assuming both l_0 and p_1 are nonzero) corresponds to

$$\operatorname{sgn}(p_1) = \operatorname{sgn}(l_0), \tag{23}$$

and

$$l_0 p_1 \leq 2. \tag{24}$$

Robustness (2)

– Zero converged error is therefore guaranteed using only knowledge of the sign of p_1 and an upper bound on $|p_1|$.

– Perturbations on p_2, p_3, etc. do not destabilize the ILC system.

– (24) can be satisfied for an arbitrarily large upper bound on $|p_1|$ by choosing $|l_0|$ sufficiently small: ILC is stably robust to all perturbations that do not change the sign of p_1.

– Robustness related to monotonic convergence is another issue.

Robustness (3)

- Consider the following uncertain system:

$$P(q) = \hat{P}(q)[1 + W(q)\Delta(q)], \quad \text{[redacted]} \tag{25}$$

where $\hat{P}(q)$ is the nominal system model, $W(q)$ is known and stable, and $\Delta(q)$ is unknown and stable with $\|\Delta(z)\|_\infty < 1$.

- **THEOREM 5**

If

$$|W(e^{j\theta})| \leq \frac{\gamma^* - |Q(e^{j\theta})||1 - e^{j\theta} L(e^{j\theta})\hat{P}(e^{j\theta})|}{|Q(e^{j\theta})||e^{j\theta} L(e^{j\theta})\hat{P}(e^{j\theta})|}, \tag{26}$$

for all $\theta \in [-\pi, \pi]$, then the ILC system (1), (4), and (25) with $N = \infty$ is monotonically convergent with convergence rate $\gamma^* < 1$.

Robustness (4)

- The monotonic robustness condition depends on the dynamics of $P(q)$, $Q(q)$, and $L(q)$, not only on the first Markov parameters p_1 as for the stability robustness.
- The most direct way to increase robustness at a given θ is to decrease the Q-filter gain $|Q(e^{j\theta})|$, yielding a trade off with performance!

Robustness (5)

- Robust performance: what is the effect of noise, nonrepeating disturbances, and initial condition variation on performance?
- These effects are introduced by adding an iteration dependent exogenous signal d_j to (1).
- Ideally we do not want the ILC to attempt to learn from d_j, so we might expect that slower learning is desirable. However, it has been shown that fast learning is required at the frequencies where d_j is large in case d_j is a stochastic disturbance, that is

$$|Q(e^{j\theta})[1 - e^{j\theta}L(e^{j\theta})\hat{P}(e^{j\theta})]| << 1$$

  at these frequencies.
- In order to realize this, $|Q(e^{j\theta})|$ should be made small at these frequencies, that is trade off between nominal and robust performance.

Design

From a practical perspective, the goal of ILC is to generate an open-loop signal that approximately inverts the system's dynamics to track a reference signal or reject a repeating disturbance. Three popular ILC algorithms and design techniques are discussed:

- PD-type and tunable designs
- Plant inversion methods

PD-type and tunable designs

- Most popular approach since it is a tunable design that does not require an accurate model of the system, similar to PID feedback control.
- **P** and **D** actions are in most cases present, not an **I** action because ILC has a natural integrator action from one trial to the next.
- The PD-type learning function can be written as:

$$u_{j+1}(k) = u_j(k) + k_p e_j(k+1) + k_d(e_j(k+1) - e_j(k)), \tag{27}$$

  where k_p is the proportional gain and k_d is the derivative gain.
- Based on theorem 1, one can state that AS is guaranteed if $|1 - (k_p + k_d)p_1| < 1$. So if p_1 is known, it is always possible to find a k_p and k_d such that the ILC system is AS.

PD-type and tunable designs (2)

- Monotonic convergence is not always possible with PD-type ILC.
- Most favorable and generally applicable approach to achieving monotonic convergence is to include a *low-pass* Q-filter: it disables learning at higher frequencies (good for monotonic convergence) which has the additional benefits of added robustness and high-frequency noise filtering.

PD-type and tunable designs (3)

- Tuning of Q-filter, k_p, and k_d: guidelines
  - First select a filter type and order, and use the bandwidth of the filter as the tuning variable.
  - For each set of k_p and k_d, the learning is reset and run for sufficient iterations to determine transient behavior and asymptotic error.
  - Start with low gains and bandwidth.
  - In general, the learning gains influence the rate of convergence, the Q-filter influences the converged error performance.
  - Increasing Q-filter bandwidth decreases robustness but increases performance.

PD-type and tunable designs (4)

- Two-step tuning method based on the AS and the monotonic convergence condition (15):

$$|1 - e^{j\theta} L(e^{j\theta}) P(e^{j\theta})| < \frac{1}{|Q(e^{j\theta})|},$$

  for all $\theta \in [\pi, \pi]$.

  1. Using the Nyquist plot of $e^{j\theta} L(e^{j\theta}) P(e^{j\theta})$, the learning gains are tuned to maximize the range $\theta \in [0, \theta_c]$ over which $e^{j\theta} L(e^{j\theta}) P(e^{j\theta})$ lies inside the unit circle centered at 1.
  2. The Q-filter bandwidth is selected last to satisfy the stability condition.

Plant inversion methods

- Use a model of the inverted system dynamics in the ILC algorithm:

$$u_{j+1}(k) = u_j(k) + \hat{P}^{-1}(q)e_j(k).$$

- This corresponds to (4) with $Q(q) = 1$ and $L(q) = q^{-1}\hat{P}^{-1}(q)$, which is causal.
- If $\hat{P}(q)$ is an exact model model of the system and the inversion of this model does not introduce errors, it can be verified from theorems 4 and 5 that the convergence rate $\gamma = 0$: the convergence occurs in just one iteration and $e_\infty = 0$.
- Problem: nonminimum phase systems results in an unstable filter: use a stable inversion approach that results in a noncausal learning filter.

Plant inversion methods (2)

- In case of minimum phase systems: the success depends on the accuracy of the model: mismatch between model $\hat{P}(q)$ and actual system $P(q)$ prevents convergence from occurring in one iteration, and can even lead to poor transient behavior.

- Considering an uncertain system with multiplicative uncertainty (25), and based on theorem 6, monotonic convergence with rate better than γ^* occurs if $|W(e^{j\theta})| < \gamma^*$ for all $\theta \in [-\pi, \pi]$.

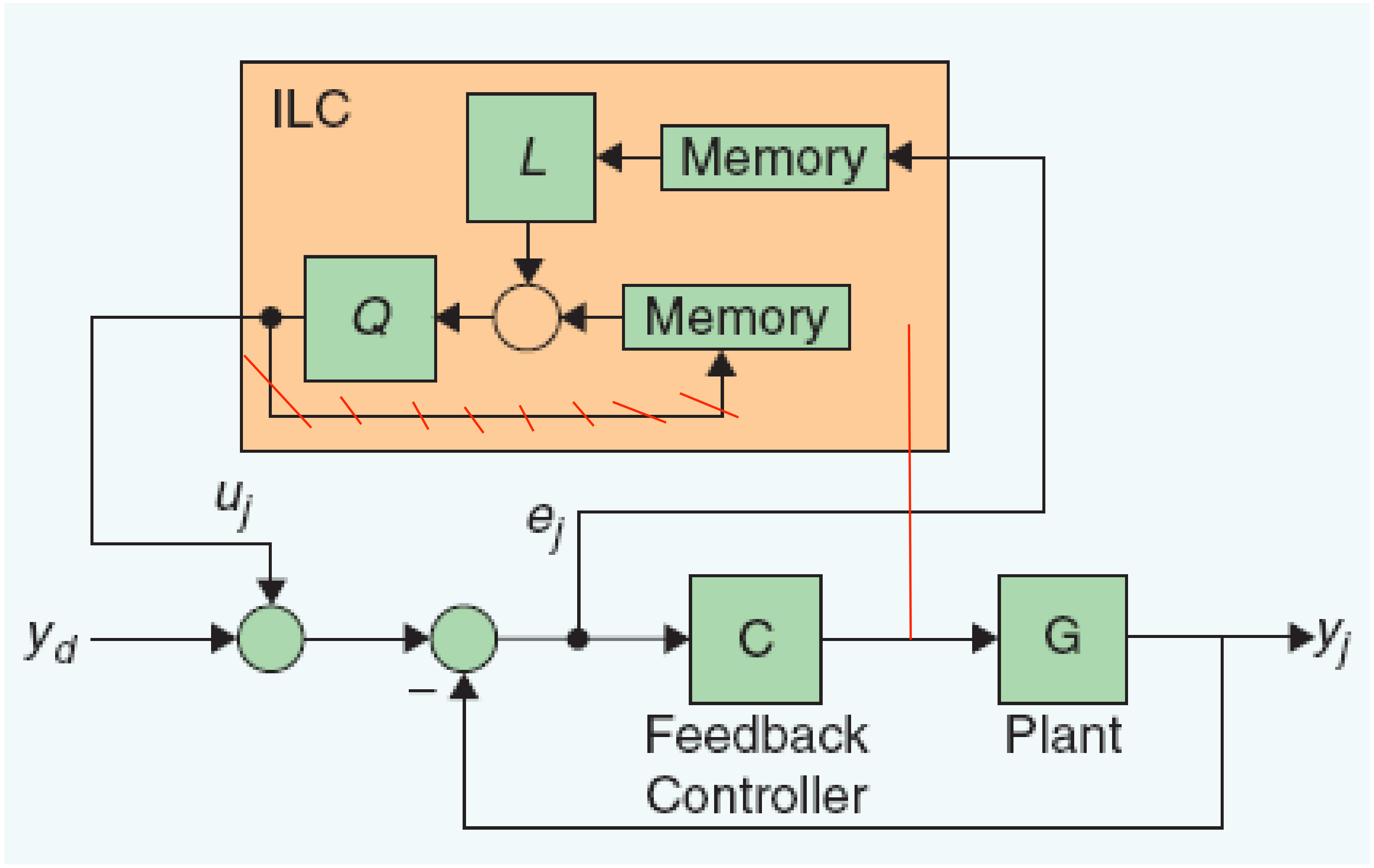
- If at a certain frequency θ_0, $|W(e^{j\theta_0})| > 1$, signifying an uncertainty greater that 100%, then the ILC system will not be robustly monotonically convergent.

- Large uncertainty typically occurs at high frequencies: remedy here is to include a lowpass Q-filter.

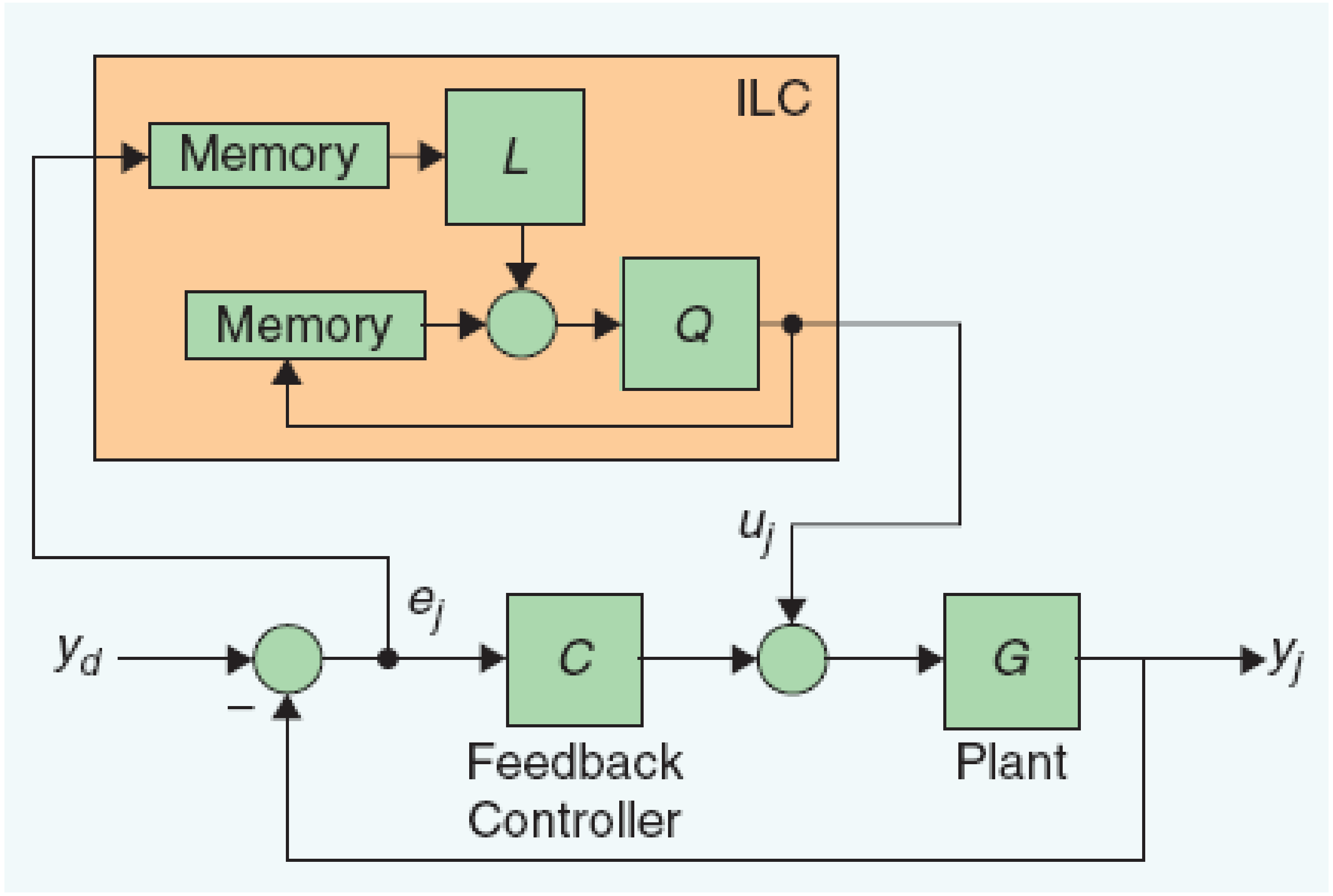
Using feedback control with ILC

- As presented here: ILC uses open-loop control action only, which cannot compensate for nonrepeating disturbances.
- In most applications, a well-designed feedback controller must be used in combination with ILC.
- ILC can be combined with a feedback controller in two ways, as shown on the next slide: *serial* and *parallel* arrangement.

- serial arrangement



- parallel arrangement



- Serial arrangement: alters the reference signal to the system. This concept is useful when applying ILC to a preexisting systems that uses a controller that does not allow access to modifying the control signal to the plant.

$$y_j = \underbrace{(1+GC)^{-1}GC}_{P}\, u_j + \underbrace{(1+GC)^{-1}GCy_d}_{d}$$

- Parallel arrangement: alters directly the control signal to the plant. This arrangement is more intuitive to motion control designers who add feedforward signals directly to the control signal for improved tracking performance. As the ILC converges, feedback controller applies less effort.

$$y_j = \underbrace{(1+GC)^{-1}G}_{P}\, u_j + \underbrace{(1+GC)^{-1}GCy_d}_{d}$$

Causal and noncausal learning

- The ILC algorithm (4) is causal if $u_{j+1}(k)$ depends only on $u_j(h)$ and $e_j(h)$ for $h \leq k$.
- The ILC algorithm (4) is noncausal if $u_{j+1}(k)$ is also function of $u_j(h)$ and $e_j(h)$ for some $h > k$. It is implementable in practice because the entire time sequence of data is available from all previous iterations.
- Consider a simple noncausal ILC algorithm

$$u_{j+1}(k) = u_j(k) + k_p e_j(k+1).$$

and a causal one

$$u_{j+1}(k) = u_j(k) + k_p e_j(k).$$

A disturbance $d(k)$ enters the error as $e_j(k) = y_d(k) - P(q)u_j(k) - d(k)$. So, the noncausal algorithm can anticipate the disturbance $d(k+1)$ and preemptively compensate with the control $u_{j+1}(k)$.

- Feedback equivalences exist for causal ILC algorithms, so they have little value ... this is not the case for noncausal ILC algorithms.

Example: Plant inversion approach

- Plant and model description
- Plant inversion assuming no model mismatch
- Unstable ILC due to model mismatch
- Stabilized ILC by adding Q-filter

Plant and model description

- Consider the following 4th order system:

$$P_p(z) = \frac{B_p(z)}{A_p(z)} = \frac{10^{-3}(0.2087z^3 - 0.2047z^2 - 0.1932z + 0.2038)}{z^4 - 3.5687z^3 + 5.0765z^2 - 3.4467z + 0.9390}$$

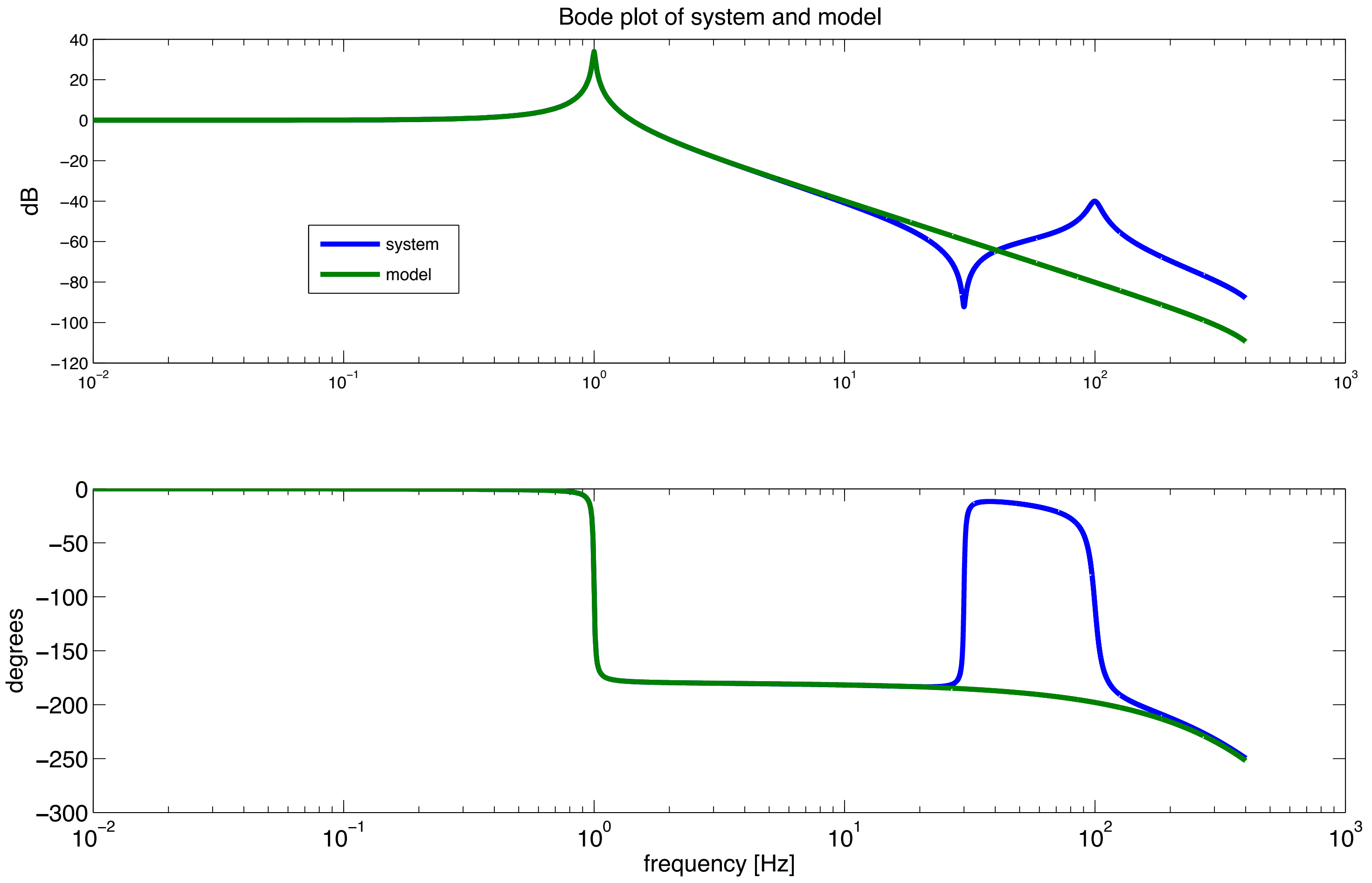
  and an approximate 2nd order model:

$$P_m(z) = \frac{B_m(z)}{A_m(z)} = \frac{10^{-4}(0.1974z + 0.1974)}{z^2 - 1.9998z + 0.9999}$$

- discrete time system and model, with sampling frequency $1000Hz$
- two complex conjugated poles at $1Hz$ with $\zeta = 0.01$, and at $100Hz$ with $\zeta = 0.05$, and two complex conjugated zeros at $30Hz$ with $\zeta = 0.01$.

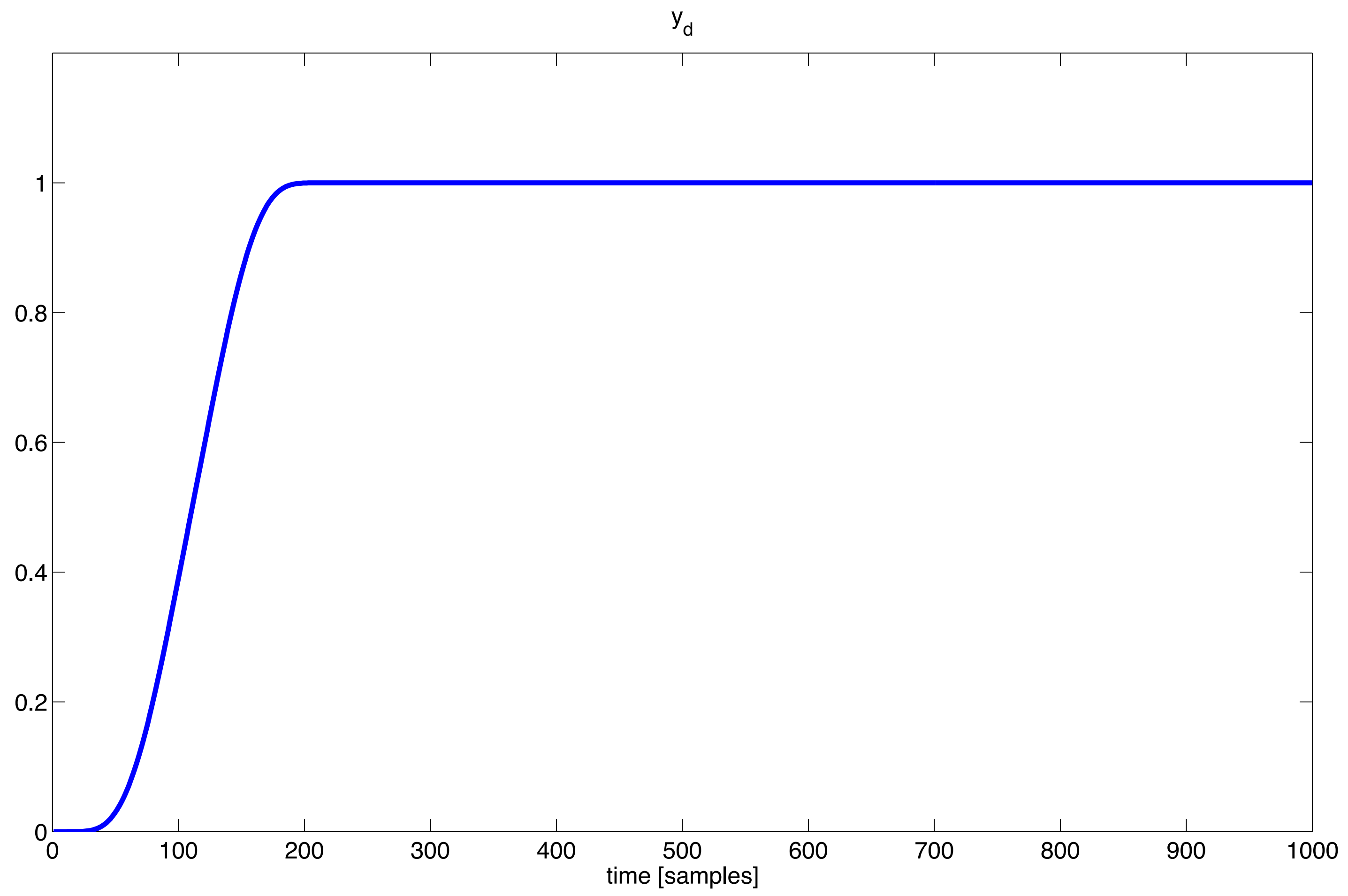
Plant and model description (2)

- Bodeplot of plant and model



Plant and model description (3)

- Desired output trajectory y_d



Plant inversion assuming no model mismatch

- Assume that model and plant are equal, and 2nd order:

$$\begin{aligned}
P_m(z) &= P_p(z) = \frac{B_p(q^{-1})}{A_p(q^{-1})} = \frac{B_m(q^{-1})}{A_m(q^{-1})} \\
&= \frac{10^{-4}(0.1974z + 0.1974)}{z^2 - 1.9998z + 0.9999} \\
&= \frac{b_1 q^{-1} + b_2 q^{-2}}{1 + a_1 q^{-1} + a_2 q^{-2}}
\end{aligned}$$

- ILC algorithm equals:

$$\begin{aligned}
u_{j+1}(k) &= u_j(k) + \frac{1 + a_1 q^{-1} + a_2 q^{-2}}{b_1 q^{-1} + b_2 q^{-2}} e_j(k) \\
&= u_j(k) + \underbrace{\frac{1 + a_1 q^{-1} + a_2 q^{-2}}{b_1 + b_2 q^{-1}}}_{L(q)} e_j(k+1)
\end{aligned}$$

Plant inversion assuming no model mismatch (2)

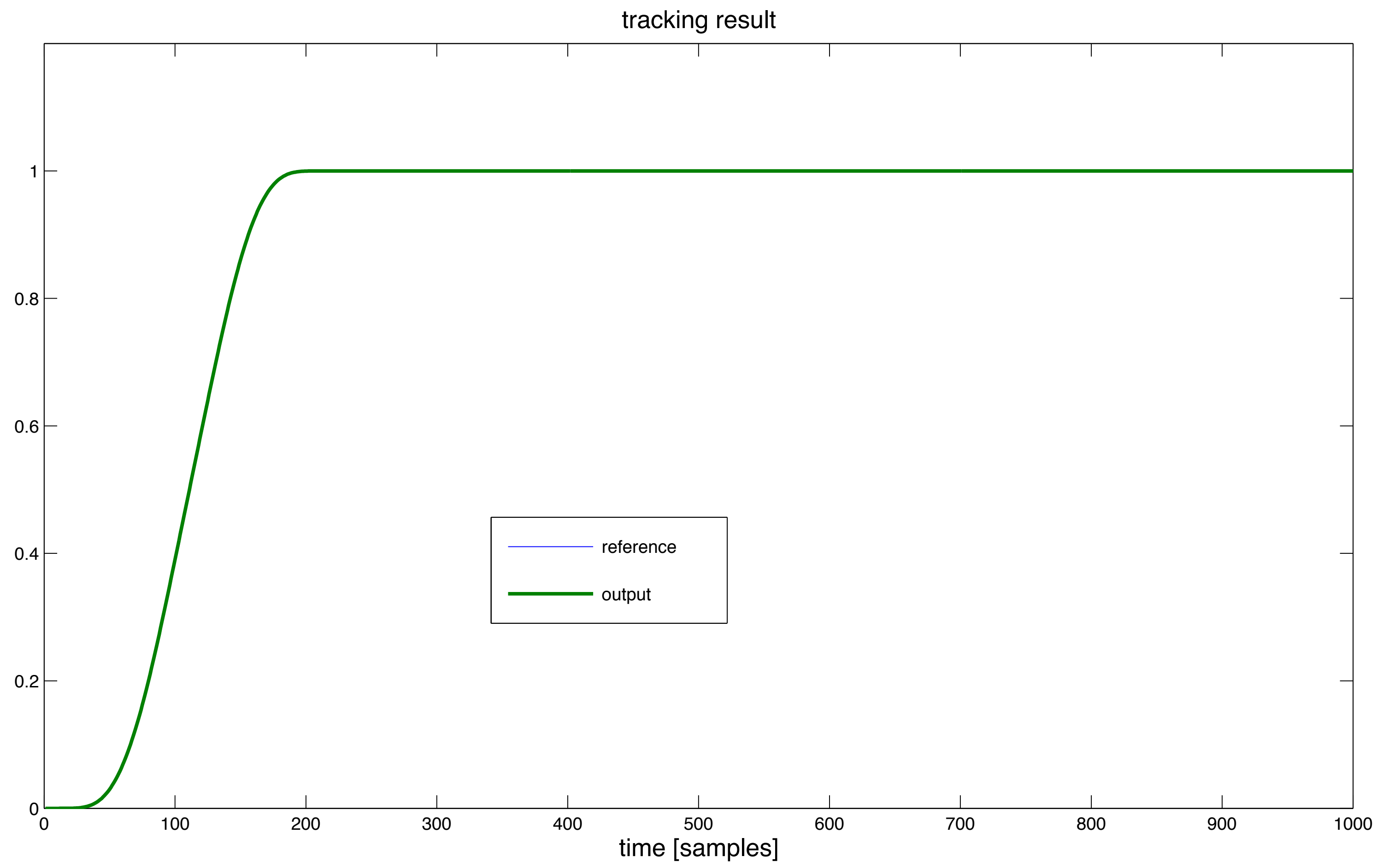
- With $Q(q) = 1$, this yields for the stability criterion (theorem 2):

$$\|Q(z)(1 - zL(z)P_m(z))\|_\infty = 5\,10^{-16} << 1$$

- Perfect tracking is obtained after one iteration.

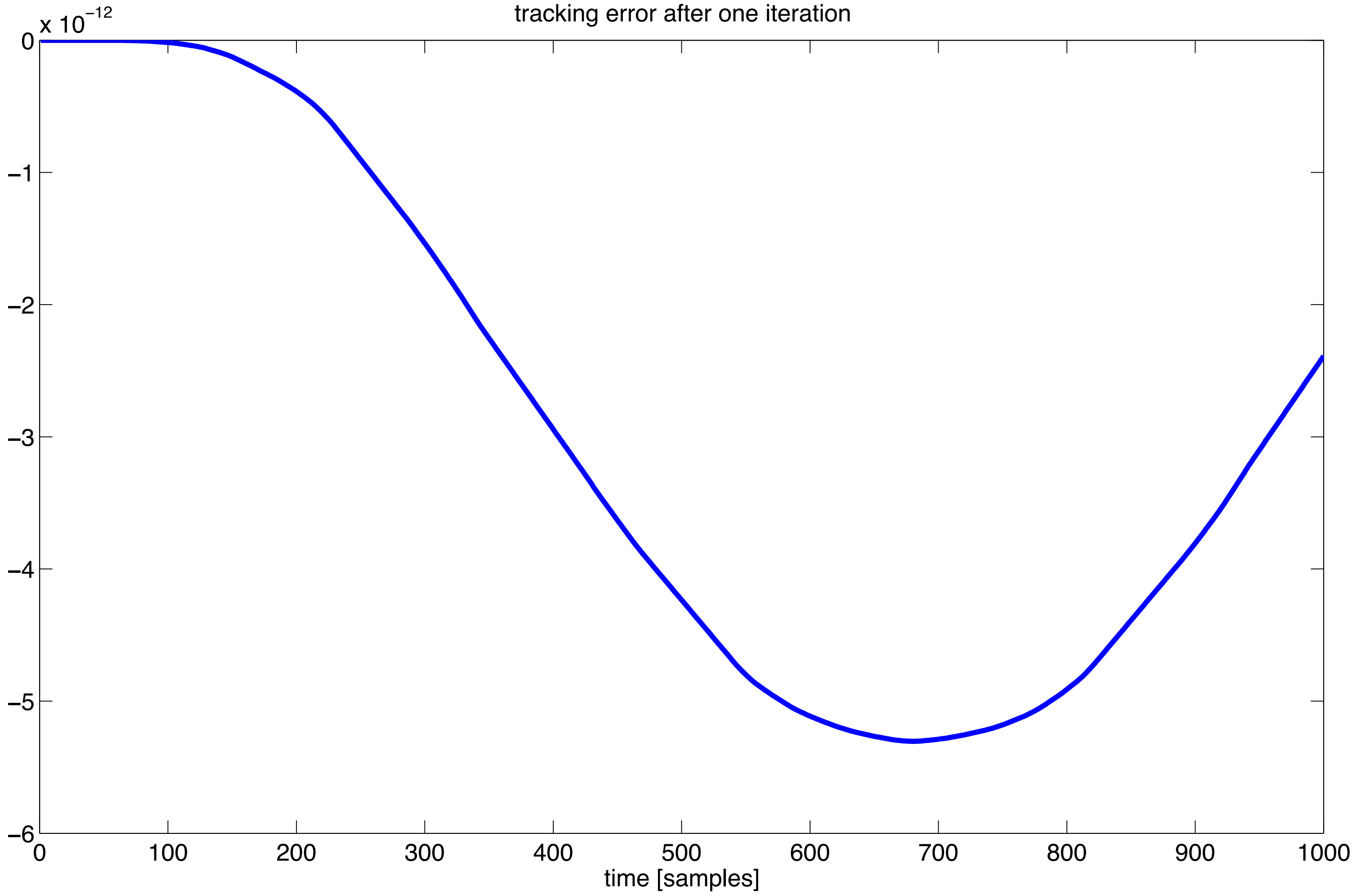
Plant inversion assuming no model mismatch (3)

- Tracking result after one iteration



Plant inversion assuming no model mismatch (4)

- Tracking error after one iteration

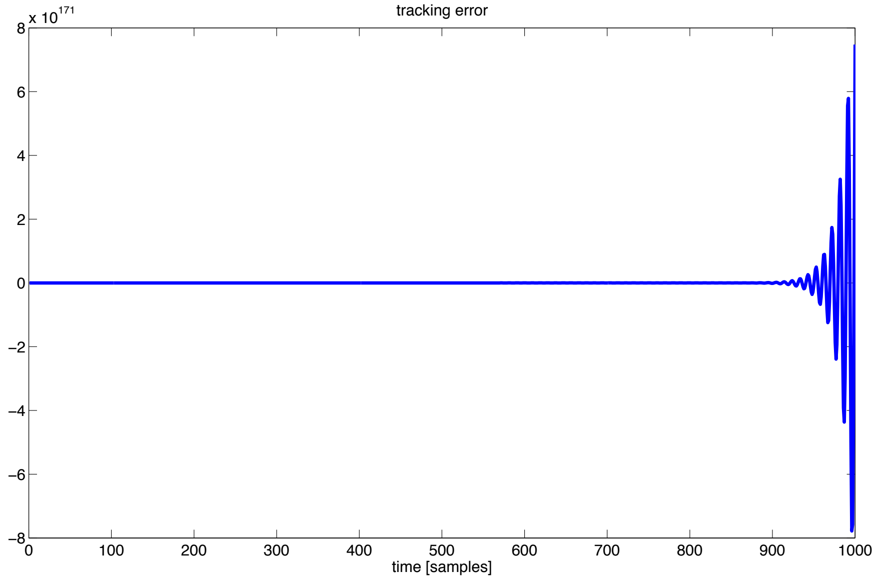


Unstable ILC due to model mismatch

- Consider the same ILC algorithm (with $Q = 1$) based on the inverse of the second order model, and apply it to the fourth order model
- ILC is unstable:

$$\|Q(z)(1 - zL(z)P_p(z))\|_\infty = 5661 >> 1$$

- Tracking error after 100 iterations

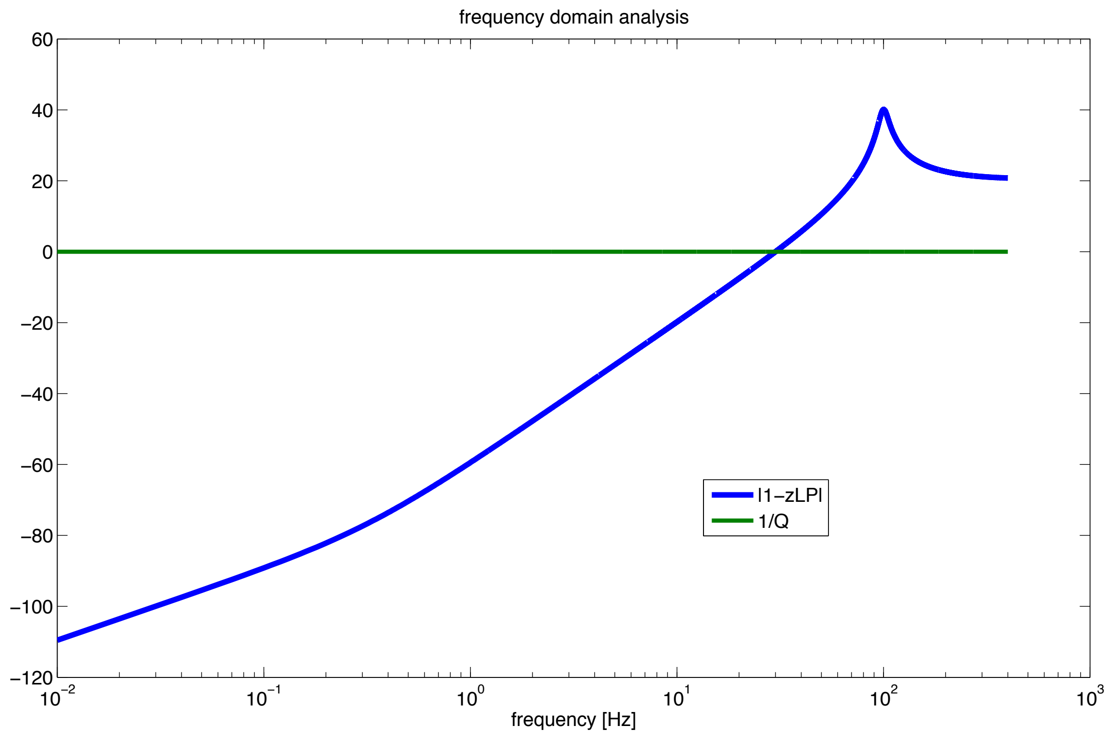


Unstable ILC due to model mismatch (2)

- Analysis in frequency domain: check if:

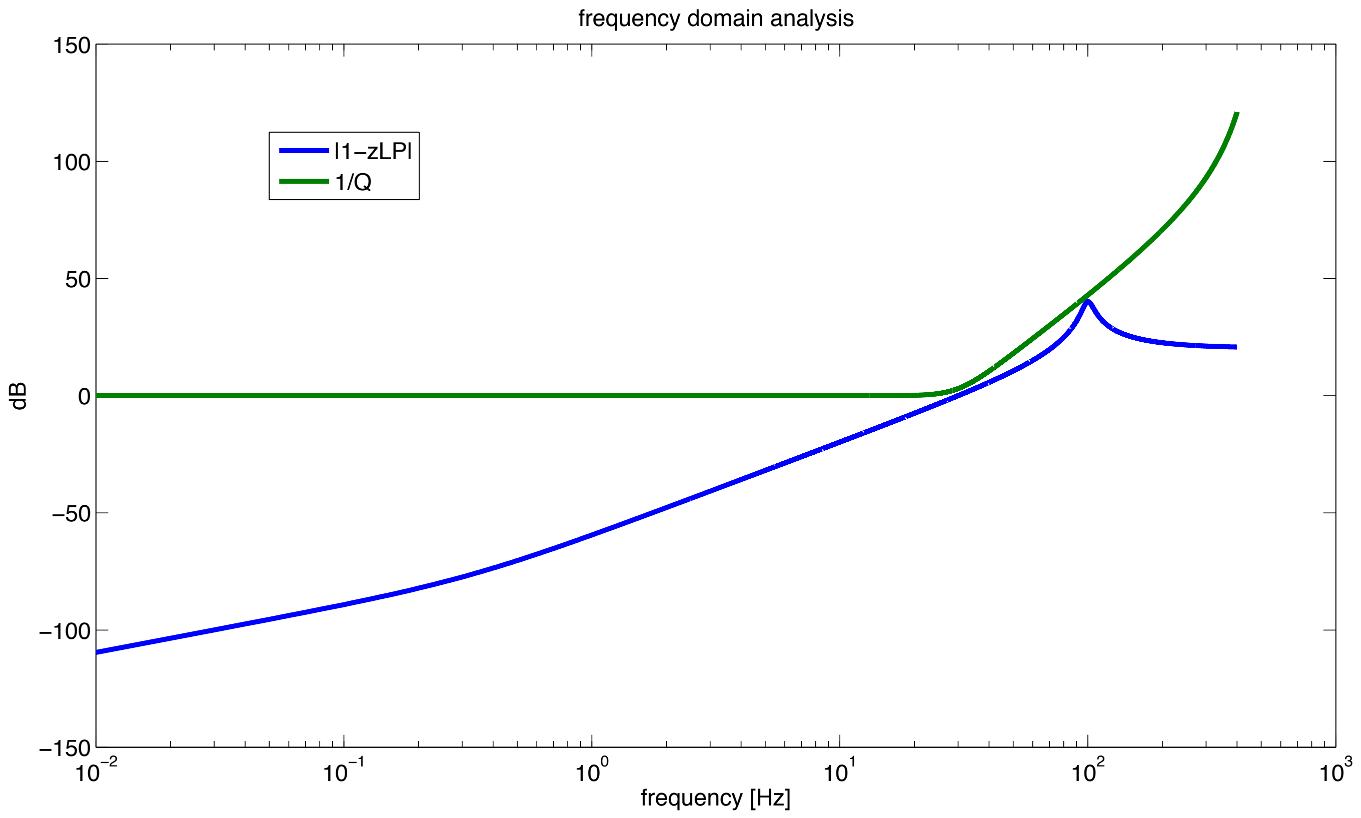
$$|1 - e^{j\theta} L(e^{j\theta}) P_p(e^{j\theta})| < \frac{1}{Q(e^{j\theta})}$$

for all $\theta \in [\pi, \pi]$



Stabilized ILC by adding Q-filter

- Add a lowpass Q-filter such that above mentioned condition is satisfied: fourth order butterworth filter with cut-off frequency at $30Hz$.



Stabilized ILC by adding Q-filter (2)

- The ILC

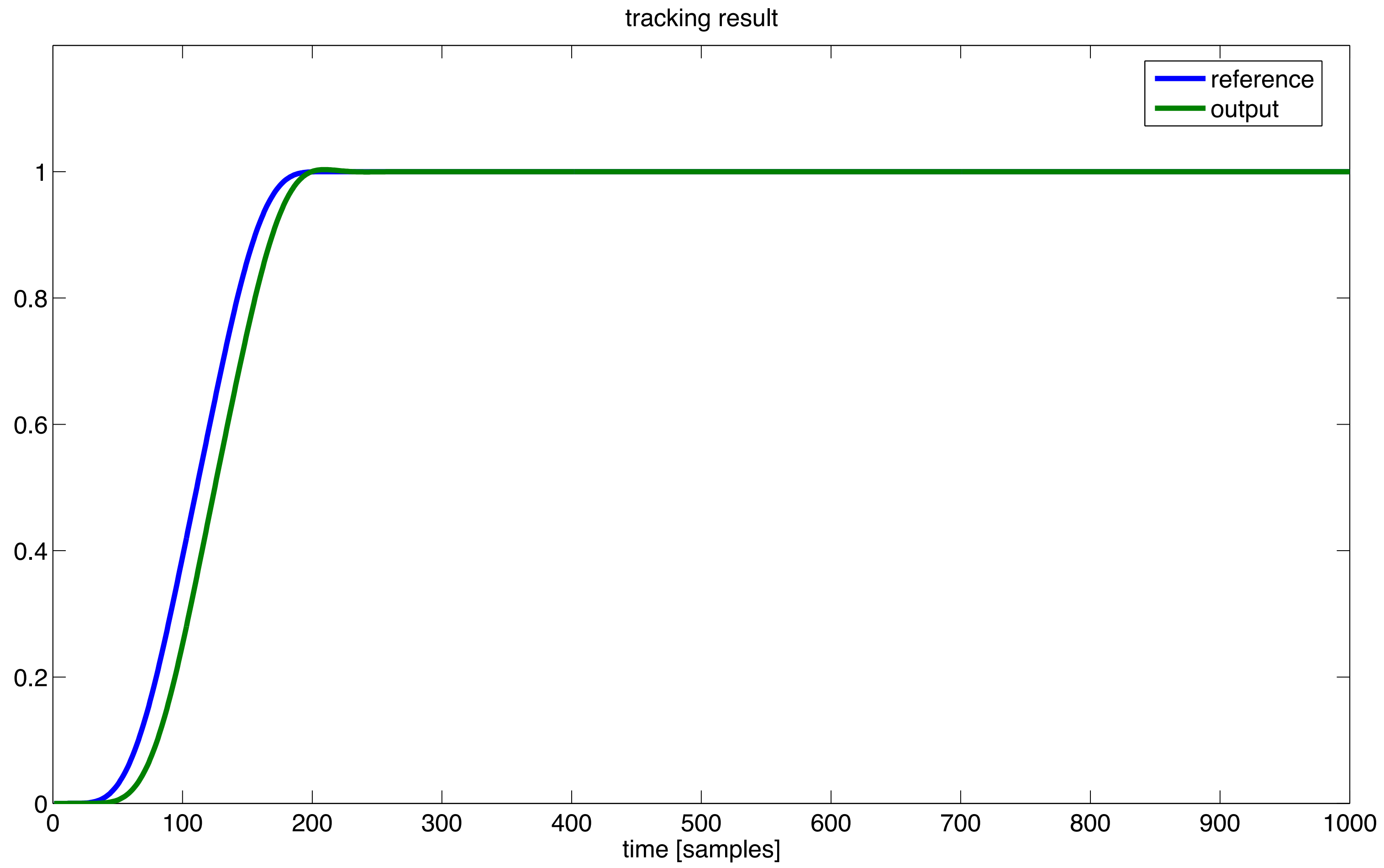
$$u_{j+1}(k) \quad = \quad Q(q)\left(u_j(k) + \frac{1 + a_1 q^{-1} + a_2 q^{-2}}{b_1 q^{-1} + b_2 q^{-2}} e_j(k)\right).$$

is stable:

$$\|Q(z)(1 - zL(z)P_p(z))\|_\infty = 0.73 < 1$$

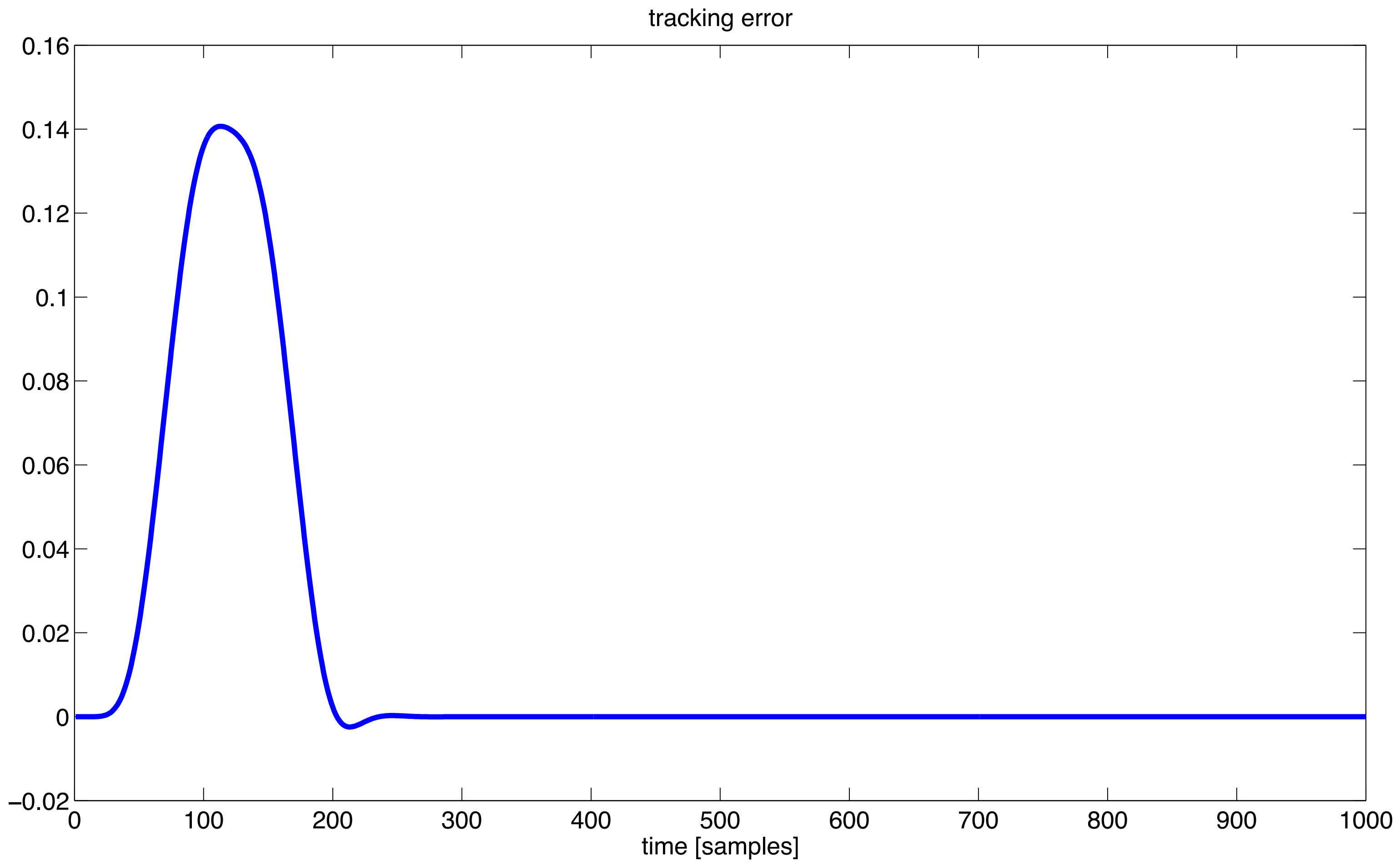
Stabilized ILC by adding Q-filter (3)

- Tracking result after 10 iterations:



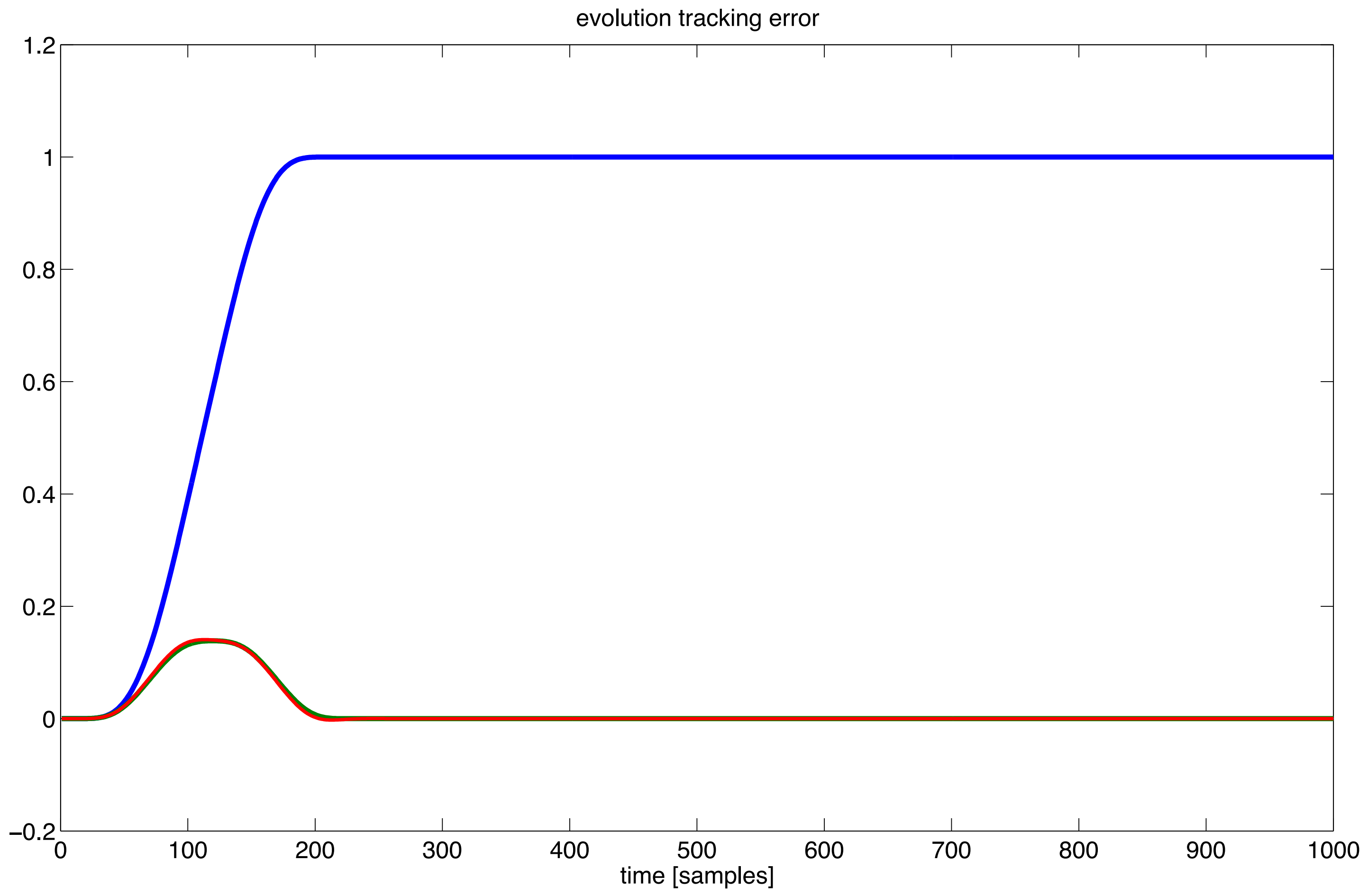
Stabilized ILC by adding Q-filter (4)

- Tracking error result after 10 iterations:



Stabilized ILC by adding Q-filter (5)

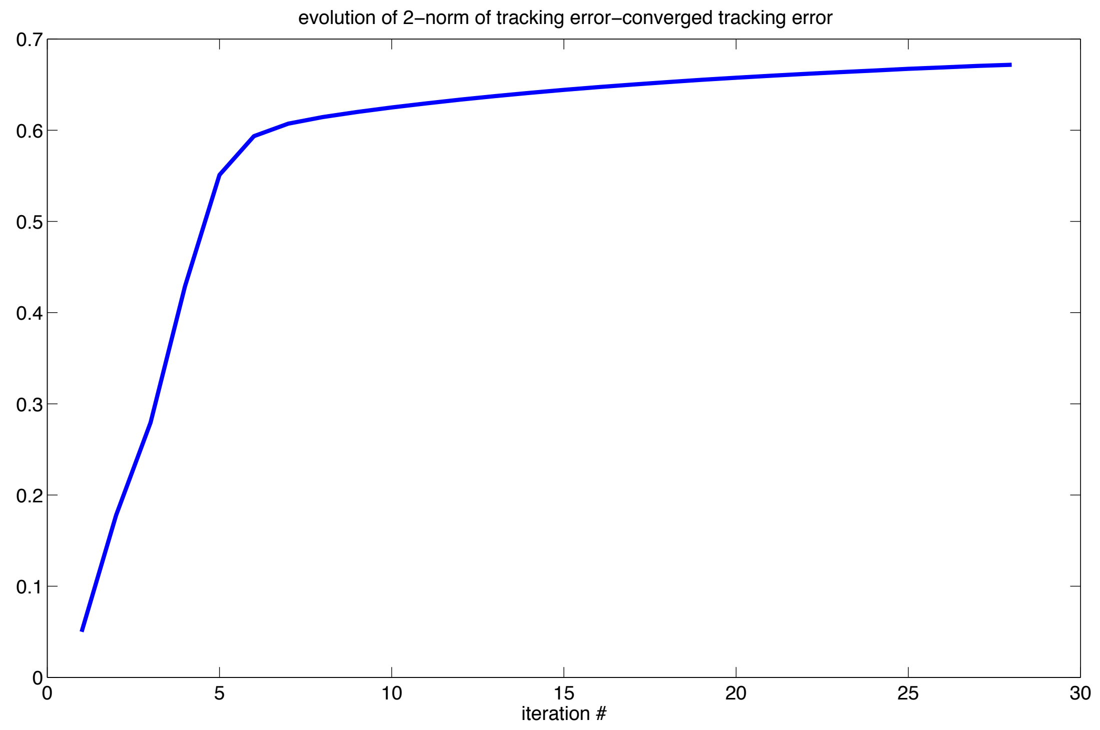
- Evolution of the tracking error:



Stabilized ILC by adding Q-filter (6)

- Evolution of 2-norm of the difference between the tracking error and converged tracking error:

$$\frac{\|e_\infty - e_{j+1}\|_2}{\|e_\infty - e_j\|_2} < \gamma_2 = 0.73$$



Concluding remarks

- Performance can be improved with ILC for repeating tasks and disturbances.
- Different design methods exist.
- Important issues: stability, (monotonic) transient learning behavior, robustness.
- Ongoing research: ILC for nonlinear systems, influence of noise and nonrepeating disturbances, ILC for similar systems, tasks, and disturbances.