Iterative Learning Control and Applications in Rehabilitation

Ying Tan

The Department of Electrical and Electronic Engineering, School of Engineering
The University of Melbourne
1. A brief introduction of the University of Melbourne

2. A brief introduction of my research interests.

3. Iterative learning control (ILC)

4. Applications in rehabilitations: rehabilitation robotic systems and human motor systems

5. Multi-loop ILC motivated from applications
1. The University of Melbourne was established in 1853.
2. It is one of top universities in Australia.
3. It has 6500 full and part-time staff and a student body of more than 47,000, including more than 11,800 international students from over 120 countries.
1. The School of Engineering was founded in 1861.

2. It has 6 departments:

3. It has 2700 students in coursework programs (nearly 700 at the Masters level) and more than 500 students in research training.

4. 400 teaching and research staff.
Ranking the 15th around the world (Electrical engineering ranking 2011)

5 IEEE Fellows (7 altogether, 2 left)

3 Fellow of Australian Academy of Science

3 Fellow of Australian Academy of technological Science and Engineering

4 main research groups: Signals and Systems; Communication; Photonics; Bio-engineering.
My research interests

- On-line optimization using extremum seeking.
- Formation control (leader-follower consensus).
- Sampled-data of distributed parameter systems.
- Stability analysis of nonlinear time-varying systems.
- Control of maglev train systems.
- Intelligent systems and control (Iterative Learning Control).
A motivating example: a tidal turbine.

- flowing water
- blades
- spinning
- energy (power)
A motivating example: a tidal turbine.

\[ P \approx C_p \cdot 0.5 \cdot \rho \cdot A \cdot V^3 \]

- \( P \): the power generated (in watts);
- \( C_p \): the turbine coefficient of performance;
- \( \rho \): the density of the water (seawater is 1025 kg/m³);
- \( A \): the sweep area of the turbine (in m²);
- \( V^3 \): the velocity of the flow cubed
A motivating example: a tidal turbine.

Assume that the turbine can rotate with $\theta$ such that

$$A = f(\theta)$$

unknown.

$$\Rightarrow P \approx C_p \cdot 0.5 \cdot \rho \cdot f(\theta) \cdot V^3$$

It is desired to find $\theta^*$ such that $P$ is maximized.

Extremum seeking can find $\theta^*$ by using the measurements of $P$ and a special mechanism.
Extremum Seeking Control

A diagram of ESC

The system $Q(\cdot)$ is unknown.

$$\exists y^*, s.t. \quad y^* = \max_{\theta \in R} Q(\theta) = Q(\theta^*)$$

Control objective: finding $\theta^*$ such that

$$\lim_{t \to \infty} y(t) \to y^*$$
Some results obtained:

- Various global ESC algorithms
- The performance analysis of ESC in terms of the choice of the dither signals
- Applications in bio-reactors
- A unified ESC design framework
- ESC with input saturation
What is an iterative learning control?

It is a method of tracking control for systems that work in a repetitive mode.

The motivation of ILC: is to get a better \textit{transient} response.

A standard control loop

Find a controller (feedback), such that:

The controller can ensure stability and zero steady-state error.
The key idea is to use the repetition of the system to improve the transient response.

This is just like “human-learning”. We learn through the experience.

The control input uses (error information) obtained last trial to predict and “correct”.

![Iterative Learning Control Diagram]
Applications of ILC

- Robotic systems;
- Batch processes;
- High precision CNC machining;
- Hard disc drive;
- Milling and laser cutting
- Traffic flow control;
- Rehabilitation
A simple case: a nonlinear time-varying discrete-time plant

\[ x_i(k+1) = f(k, x_i(k), u_i(k)) \]
\[ y_i(k) = h(k, x_i(k), u_i(k)), \quad k = 0, 1, \cdots N \]

Control objective: for a desired trajectory \( y_d(k) \) find a sequence of input \( u_i(k) \), such that

\[ \lim_{i \to \infty} |e_i(k)| = 0, \forall k = 0, 1, \cdots N \]

\[ e_i(k) = y_d(k) - y_i(k) \]
A possible learning mechanism (1)

The simplest one:

Iterative Learning Control

\[ u_{i+1}(k) = u_i(k) + g(e_i(k), e_{i-1}(k), \ldots, e_{i-1}(k-n)), n \leq N \]

\[ e_i(k) = y_d(k) - y_i(k) \]

\[ u_{i+1}(k) = u_i(k) + qe_i(k) \quad k \in \{0,1,\ldots,N\} \]
A possible learning mechanism (2): non-causal one.

It is possible to design ILC algorithm to ensure that the tracking error converges (along the iteration domain) with a very limited knowledge of the plant (learn the plant by repetitions).
Common features:

- The same control task.
- The same duration

\[ k \in \{0,1,\cdots,N\} \]

\[ \lim_{i \to \infty} |e_i(k)| = 0, \]

\[ \forall k = 0,1,\cdots,N \]
There is a hybrid nature in ILC: continuous in finite-time domain (most real plants are continuous-time) and discrete in iteration domain.

There are not many effective ways to analyze the properties of ILC. Tools are relatively simple.

There are three major tools: contraction mapping based ILC (CM-ILC), 2D ILC and composition energy function based ILC (CEF-ILC).
CM-ILC: focus on input-output mapping, ignoring the dynamics of plant.

\[
\begin{align*}
\dot{x}_i(t) &= f(t, x_i(t), u_i(t)) \\
y_i(t) &= h(t, x_i(t), u_i(t)), \quad t \in [0,T]
\end{align*}
\]

\(f(t, x, u)\) and \(h(t, x, u)\) are \(\text{GLC}\) with respect to \((x, u)\), uniformly in \(t\)

\[
0 < \alpha_1 \leq \frac{\partial h}{\partial u}(t, x, u) \leq \alpha_2 \quad \forall x, \forall t \in [0,T]
\]
By design a sequence of input signal \( \{u_i(t)\}_{i \in N} \) such that

\[
\|e_{i+1}\|_\lambda \leq \rho \|e_i\|_\lambda, \rho \in (0,1)
\]

\[
\|e_i\|_\lambda = \max_{t \in [0,T]} e^{-\lambda t} |e_i(t)|, \quad \lambda > 0
\]

When \( \lambda \) is sufficiently large, the dynamics of the system is ignored and the output mapping will dominate.
2D-ILC: mainly working for discrete-time plants.

\[ x_i(k+1) = f(k, x_i(k), u_i(k)) \]
\[ y_i(k) = h(k, x_i(k), u_i(k)), \quad k = 0, 1, \ldots, N \]

Most results are focused on linear-time-invariant systems. Not many nonlinear results are available.
Iterative Learning Control

CEF-ILC: using the concept of composite energy function: including energy along time and energy along iteration as well.

It can work well for both continuous-time plants and discrete-time plants.

\[ E_i(t, x_i, u_i) \geq 0, t \in [0, T] \]

\[ E_i(t, x_i, u_i) - E_{i-1}(t, x_{i-1}, u_{i-1}) \leq -\alpha\|e_i(t)\|, t \in [0, T] \]

\[ \lim_{i \to \infty} |e_i(t)| = 0, \]
However, a systematic development of CEF-ILC is still not available.

Compared with Lyapunov methods widely used in nonlinear control systems (in time domain only), there are not many results in constructing CEF for ILC.

It is non-trivial to extend some well-known results such as small-gain theorem in nonlinear control systems to CEF-ILC.
ILC control is now used in rehabilitation area, especially in the area of rehab after stroke.

Rehabilitation of sensory and cognitive function typically involves methods for retraining neural pathways or training new neural pathways to regain or improve neuro-cognitive functioning that has been diminished by disease or trauma.
Stroke is Australia’s second single greatest killer after coronary heart disease and a leading cause of disability.

The number of stroke survivors in Australia is over 406,700. That is one stroke every 10 minutes.

About 88 per cent of stroke survivors live at home and most have a disability.

Strokes cost Australia an estimated $2.14 billion a year. Of those who survive stroke with severe paralysis, only 5% regain upper limb function.
We are doing two different types of work for rehab:

- One is to build a rehab-robotic system to help patient to recover;
- The other is try to understand how human motor system (HMS) works for health people by constructing a computation model of human motor systems.

These two types of work are all closely related to iterative learning control as the key concept of rehab is “learning/practising promotes recovery”
I am now working with Chris Freeman from New Southampton University improve the rehabilitation robotic systems.
Applications on Rehab

EMMA DOUGLAS
Southampton

south.today@bbc.co.uk
The multi-loop ILC algorithms are needed in this project.

1. Learning a personalized trajectory for each patient
2. Learning the muscle model (or dynamics) of each patient
3. Design ILC for a given task

Three loops are inter-connected. The time-scale may be different.
What is HMS?

HMS describes the capability of an individual to calibrate the characteristics of his body so as

- To produce the desired motion.
- To adjust to the dynamics of the environment.
Why a computation model of HMS is needed

It can provide better understanding of individual human learning. This will lead to better diagnosis of individual patients suffering from abnormal behaviour such as after-stroke patients.

Building such computation model needs well-designed experiments and experiments data
We have build a computation model for HMS when performing a reaching task in time-varying environment.
Computation Model of HMS
Result of experiments
Computation model from ILC (ILC updates the model from output). Simulation results from computation model are:
Simulation results from computation model of HMS match well to experimental results.

This shows the computation model obtained captures the learning ability of human beings.

HMS has a hierarchical structure of three interconnected processes: (1) motor planning, (2) motor control and (3) motor execution.

Therefore, multi-loop ILC design is also needed.
Multi-loop ILC concepts are motivated from various applications.

The work of multi-loop ILC has just started, only some preliminary results are obtained.

Here we consider only two inter-connected systems.

\[ \Sigma_2: \dot{x}_2 = f_2(t, x_2, u) \]
\[ z = h(t, x_2, u) \]

\[ \Sigma_1: \dot{x}_1 = f_1(t, x_1, z) \]
\[ y = x_1 \]
Each subsystem, there exists an ILC loop to make it work

\[ \Sigma_1 : \dot{x}_1 = f_1(t, x_1, z) \]

\[ y = x_1 \]

\[ \forall y_d(t) \in C[0,T], \exists \text{ updating law} \]

\[ z_{i+1}(t) = g_1(t, z_i(t), e_i(t)) \]

\[ s.t. \lim_{i \to \infty} |e_i(t)| = 0, \text{ point-wise} \]
Each subsystem, there exists an ILC loop to make it work

\[ \Sigma_2 : \dot{x}_2 = f_2(t, x_2, u) \]
\[ z = h(t, x_2, u) \]

\[ \forall z_d(t) \in C[0, T], \exists \text{ updating law} \]
\[ u_{i+1}(t) = g_2(t, u_i(t), \delta_i(t)) \]

\[ s.t. \lim_{i \to \infty} |\delta_i(t)| = 0, \text{ uniformly} \]
\[ \delta_i(t) = z_d(t) - z_i(t) \]
The question is how to ensure the overall system can work when these two loops are connected.

Loop 1 will provide the desired trajectory for Loop 2.

We use time-scale separation technique: Loop 1 waits until Loop 2 converges. That is: Loop 1 updates every $N$ iterations when Loop 2 already behaves well.

Then the overall system can converge to a small neighborhood of desired trajectory.
Multi-loop ILC

\[ r_{i+N_2+1}(t_s) = g_1(t_s, r_{i+N_2}(t_s), e_{i+N_2+1}(t_s)) \]

ILC Loop 1

\[ r_{i+1}(t_s) = \cdots = r_{i+N_2}(t_s) \]

\[ \delta_{i+1} \]

\[ z_{i+2}(t_s) \]

\[ z_{i+1}(t_s) \]

\[ i+1 \]

\[ i+N_2 \]

\[ i+N_2+1 \]

\[ i+2N_2 \]

\[ u_{i+1}(t_s) = g_2(t_s, u_i(t_s), \delta_i(t_s)) \]

ILC Loop 2

\[ \delta_{i+N_2+1} \]

\[ v \]
Conclusions

ILC is very useful in many engineering applications.

But the tools are not rich enough to design proper ILC schemes for complicated systems or analyze the performance.

We are trying developing more appropriate tools now.

These tools should be simple enough for engineers to use, but it is also flexible enough to cover various engineering applications.

We need work with mathematicians.