On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

presented by

Kevin L. Moore, Ph.D., P.E.
G.A. Dobelman Distinguished Chair and Professor of Engineering
Division of Engineering
Colorado School of Mines

at the

2009 ANNIE Conference
(Artificial Neural Networks in Engineering)
St. Louis, Missouri, USA
3 November 2009
Colorado School of Mines

Located in Golden, Colorado, USA
10 miles West of Denver

My back yard last week – and it’s only late October!
• In Colorado (and many places!) fishing is a popular activity
• Consider a fisherman wanting to place the bait at the right spot
Because of the refraction of light in water, the fish’s apparent location is different than its true location.
• Thus the fisherman puts the bait in the wrong place ….
  where there is not really a fish!
A skilled fisherman will use a bit of trial-and-error, eventually “learning” where to place the bait, by: casting the bait, looking at the result, changing the target location on the next cast, and repeating...
On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

- Until finally … the bait is at the right place!
On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

- The error in the result of an action …. is used to correct the action on the next try.
Iterative Learning Control (ILC)

- ILC can be described using the previous example:
  - The error in the result of an action ….
    is used to correct the action on the next try
- Now, this sounds like nothing more than feedback control
- But ILC adds a few more wrinkles:
  - The error in the sequence of results of a sequence of actions ….
    is used to correct the sequence of actions on the next try

Assume the error is relative to a reference that is the same from trial to trial
We are concerned with behavior of a system over a complete trajectory
Repetition in operation
We will be concerned with dynamic systems

On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm
Iterative Learning Control: Paradigm for processes or systems that operate in an iterative or repetitive fashion:
- Finite time or spatial interval of operation
- Return to some starting point or state between repetitions
- Improve the performance from one operation to the next.

Such systems are known as
- Iterative
- Trial-to-trial
- Multi-pass
- Repetitive
- Periodic
- Batch-to-batch
- Run-to-run
- Pass-to-pass
- ...

Such systems arise in a variety of applications, including:
- Process control
- Semiconductor processing
- Robotic motion control
- Hard disk drive servo control
- Machining and manufacturing operations
- ...
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:

\[
\begin{align*}
    y_k & = f(y_{k-1}, u_k) \\
    u_{k+1} & = u_k + \gamma(y_d - y_k(t+1))
\end{align*}
\]

- A typical ILC algorithm has the form: \( u_{k+1}(t) = u_k(t) + \gamma(y_d(t) - y_k(t + 1)) \).
  - Note the “non-causal” nature of the algorithm.

- Standard ILC assumptions include:
  - Stable dynamics or some kind of Lipschitz condition.
  - Each trial has the same length.
  - System returns to the same initial conditions at the start of each trial.
  - Undefined amount of time can elapse between trials.
    - The last two features distinguish ILC from repetitive control (RC).
Example 1

- Consider the plant:

\[ y(t + 1) = -0.7y(t) - 0.012y(t - 1) + u(t) \]

\[ y(0) = 2 \]
\[ y(1) = 2 \]

- 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.
Example 1 (cont.)
Example 2

- 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

\[ x(t) = (\theta_1(t), \theta_2(t))^T \]

\[ A(x) = \begin{pmatrix} .54 + .27 \cos \theta_2 & .135 + .135 \cos \theta_2 \\ .135 + .135 \cos \theta_2 & .135 \end{pmatrix} \]

\[ B(x, \dot{x}) = \begin{pmatrix} .135 \sin \theta_2 & 0 \\ -.27 \sin \theta_2 & -.135(\sin \theta_2) \dot{\theta}_2 \end{pmatrix} \]

\[ C(x) = \begin{pmatrix} 13.1625 \sin \theta_1 + 4.3875 \sin(\theta_1 + \theta_2) \\ 4.3875 \sin(\theta_1 + \theta_2) \end{pmatrix} \]

\[ u_k(t) = \text{vector of torques applied to the joints} \]
Example 2 (cont.)

- Define the vectors:

\[
\begin{align*}
y_k &= (x_k^T, \dot{x}_k^T, \ddot{x}_k^T)^T \\
y_d &= (x_d^T, \dot{x}_d^T, \ddot{x}_d^T)^T
\end{align*}
\]

- A convergent learning controller can be defined by (Moore, et al., 1988 - related to work by Bondi, et al., 1987):

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

- \( \Gamma \) is a fixed feedback gain matrix that has been made time-varying through the multiplication by the gain \( \alpha_k \).

- \( r_k \) can be described as a time-varying reference input. \( r_k(t) \) and adaptation of \( \alpha_k \) are effectively the ILC part of the algorithm.

- With this algorithm we have combined conventional feedback with iterative learning control.
Example 2 (cont.)

![Graph showing iterative learning process](image)
Outline

• Introduction
  – Control System Design: Motivation for ILC
  – Iterative Learning Control: The Basic Idea

• On the History and Accomplishments of ILC
  – Early Work and Literature Survey
  – Selected Applications
  – Some ILC Algorithms

• An ILC Framework for the Past, Present, and Future
  – General Operator-Theoretic Framework
  – The “Supervector” Notation for Discrete-Time ILC
  – The $w$-Transform: “$z$-Operator” Along the Repetition Axis
  – ILC as a MIMO Control System
  – The Complete Framework: Iteration-Domain Variation

• Concluding Remarks: Future Research Vistas in ILC
  – Along the Same Lines: ILC for Spatial Processes (PDEs)
  – Something a Bit Different: Discrete Repetitive Processes
  – On the Far Horizon?: Iterative Learning and Extrapolation in Discrete Event Dynamic Systems
Control Engineering - History and ILC

- Prehistory of automatic control
- Primitive period
- Classical control
- Modern control
  - Classic control
  - Nonlinear control
  - Estimation
  - Robust control
  - Optimal control
  - Adaptive control
  - Intelligent control
    - H_inf
    - Interval
    - Fuzzy
    - Neural Net
    - ILC
    - ...
On the History and Accomplishments of ILC


On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

First ILC paper- in Japanese (1978)

On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

First ILC paper- in English (1984)

ILC Research History

- ILC has a well-established research history. As of the end of 2005:
  - More than 1000 papers:
    - At least four monographs (several more since)
    - Over 20 Ph.D dissertations

- But, ILC is still relatively “young” — in ieeeXplore.ieee.org, using the search term (in all fields):
  - “Iterative learning control” generates 465 papers
  - “Robust control” generates 10,075 papers
  - “Adaptive control” generates 11,654 papers
A Partial Classification of ILC Research

- Systems:
  - Open-loop vs. closed-loop.
  - Discrete-time vs. continuous-time.
  - Linear vs. nonlinear.
  - Time-invariant or time-varying.
  - Relative degree 1 vs. higher relative degree.
  - Same initial state vs. variable initial state.
  - Presence of disturbances.

- Update algorithm:
  - Linear ILC vs. nonlinear ILC.
  - First-order ILC vs. higher-order.
  - Current cycle vs. past cycle.
  - Fixed ILC or adaptive ILC.
  - Time-domain vs. frequency analysis.
  - Analysis vs. design.
  - Assumptions on plant knowledge.

- Applications: robotics, chemical processing, mechatronic systems.
ILC Research History (cont.)

Selected ILC Industrial Applications - Hard Disk Drive Servo

- Prof. YangQuan Chen, Utah State University, patents to compensate for repeatable run-out (RRO); can give up to 25% more storage; implemented as a table lookup; training is a pre-shipping calibration process (note the industrial tutorial session at 2006 CDC on this topic):

  – 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.”
Selected ILC Industrial Applications - Robotic

- Michael Norrlöf’s patent on ABB robots, US2004093119 “Path correction for an industrial robot;” used in ABB workcell for laser cutting on car frames in Chrysler’s North America vehicles.

Customers want a very high accuracy in applications such as:
- Laser cutting
- Water cutting
- Gluing
ILC in a commercial system
ILC Algorithms

- Standard iterative learning control scheme:

- **Goal**: Find a learning control algorithm
  \[
  u_{k+1}(t) = f_L(\text{previous information})
  \]
  so that for all \( t \in [0, t_f] \)
  \[
  \lim_{k \to \infty} y_k(t) = y_d(t)
  \]
Professor Arimoto’s Six Postulates of ILC:

- **P1.** Every trial (pass, cycle, batch, iteration, repetition) ends in a fixed time of duration $T > 0$.
- **P2.** A desired output $y_d(t)$ is given *a priori* over $[0, T]$.
- **P3.** Repetition of the initial setting is satisfied, that is, the initial state $x_k(0)$ of the objective system can be set the same at the beginning of each iteration: $x_k(0) = x^0$, for $k = 1, 2, \cdots$.
- **P4.** Invariance of the system dynamics is ensured throughout these repeated iterations.
- **P5.** Every output $y_k(t)$ can be measured and therefore the tracking error signal, $e_k(t) = y_d(t) - y_k(t)$, can be utilized in the construction of the next input $u_{k+1}(t)$.
- **P6.** The system dynamics are invertible, that is, for a given desired output $y_d(t)$ with a piecewise continuous derivative, there exists a unique input $u_d(t)$ that drives the system to produce the output $y_d(t)$.

Professor Arimoto proposed a learning control algorithm of the form:

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

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

Professor Arimoto also considered more general algorithms of the form:

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

• Numerous studies have considered a variation on the theme of the basic Arimoto algorithm, in both continuous and discrete-time, using **contraction mapping-based analysis and design**:

\[ u_{k+1}(t) = u_k(t) + \Gamma(s)e_k(t) \]

• Various researchers have used **gradient methods** to optimize the gain \( G_k \) in:

\[ u_{k+1}(t) = u_k(t) + G_k e_k(t + 1) \]

• **Norm-optimal ILC** works in the state space. For the plant:

\[ x_{k+1}(t + 1) = Ax_{k+1} + Bu_{k+1} \]

find \( u_{k+1} \) to minimize

\[ J = \|u_{k+1} - u_k\|^2 + \|e_{k+1}\|^2 \]

• It is also useful for analysis and design use **frequency domain-based learning control**, for example:

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

• A significant amount of work has also been done for nonlinear ILC, leading to the so-called **energy function approach**, using what are called control Lyapunov functions.

• Lots of work using **fuzzy-logic** and **artificial neural networks**.
Aside: Professor Arimoto’s Papers and Citations

- By late 2005, I categorized 45 papers by Prof. Arimoto from my personal file and on IEEE Xplore and Web of Science Citation searches. But, my data, while exhausting, is not exhaustive!

- Categorize ILC contributions as (roughly chronologically)
  - Foundations (6)
  - Robotics Foundations (6)
  - Mobile Robots (3)
  - Robustness (4)
  - Higher-Level Learning (4)
  - Learning Under Geometric Constraints (6)
  - Gripping and Multiple Manipulators (5)
  - Passivity and Impedance (11)

- In particular, the article “Bettering Operation of Robots by Learning,” *Journal of Robotic Systems*, Vol.1, No. 2, 123-140, 1984, is seminal (thanks to Dr. YangQuan Chen of Utah State University, USA, for compiling this information):
  - 474 Citations noted by Google Scholar
  - 357 citation noted by Web of Science
Outline

• Introduction
  – Control System Design: Motivation for ILC
  – Iterative Learning Control: The Basic Idea

• On the History and Accomplishments of ILC
  – Early Work and Literature Survey
  – Selected Applications
  – Some ILC Algorithms

• An ILC Framework for the Past, Present, and Future
  – General Operator-Theoretic Framework
  – The “Supervector” Notation for Discrete-Time ILC
  – The $w$-Transform: “$z$-Operator” Along the Repetition Axis
  – ILC as a MIMO Control System
  – The Complete Framework: Iteration-Domain Variation

• Concluding Remarks: Future Research Vistas in ILC
  – Along the Same Lines: ILC for Spatial Processes (PDEs)
  – Something a Bit Different: Discrete Repetitive Processes
  – On the Far Horizon?: Iterative Learning and Extrapolation in Discrete Event Dynamic Systems
Operator-Theoretic Convergence Condition

- **Theorem** (Moore, et al., 1988): 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 \( t \in [t_0, t_f] \).
  - Otherwise the error will be non-zero.

- **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 \( y_d \).
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:

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

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

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

• 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:

\[
\begin{align*}
x_k(t + 1) &= A(t)x_k(t) + B(t)u_k(t) \\
y_k(t) &= C(t)x_k(t) + D(t)u_k(t)
\end{align*}
\]

Then the same notation again results in \( Y_k = H_p U_k \), where now:

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

- 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 $\gamma$:

  – 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

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

    \[
    L = \begin{bmatrix}
    L_m & 0 & 0 & \cdots & 0 \\
    L_{m+1} & L_m & 0 & \cdots & 0 \\
    L_{m+2} & L_{m+1} & L_m & \cdots & 0 \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    L_{m+N-1} & L_{m+N-2} & L_{m+N-3} & \cdots & L_m
    \end{bmatrix}
    \]
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:

\[
L = \begin{bmatrix}
  K & K & 0 & 0 & \ldots & 0 & 0 & 0 \\
  0 & K & K & 0 & \ldots & 0 & 0 & 0 \\
  0 & 0 & K & K & \ldots & 0 & 0 & 0 \\
  \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
  0 & 0 & 0 & 0 & \ldots & K & K & 0 \\
  0 & 0 & 0 & 0 & \ldots & 0 & K & K \\
  0 & 0 & 0 & 0 & \ldots & 0 & 0 & K
\end{bmatrix}
\]
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 $\Gamma$ as follows:

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

The diagonal elements of $\Gamma$ 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
  \[ C(w) = \frac{1}{(w - 1)L} \]
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:

\[
\begin{align*}
    u_{k+1}(t) &= k_1 u_k(t) + k_2 u_{k-1}(t) + \gamma e_k(t + 1) \\
    C(w) &= \frac{\gamma w}{w^2 - k_1 w - k_2}
\end{align*}
\]

which corresponds to:
ILC as a MIMO Control System

- The term

\[ C'(w) = \frac{1}{(w - 1)}L \quad \text{or} \quad C'(w) = \frac{\gamma w}{w^2 - k_1 w - k_2} \]

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

- 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 $m = 1$):

$$u_{k+1}(t) = \bar{D}_n(z)u_k(t) + \bar{D}_{n-1}(z)u_{k-1}(t) + \cdots + \bar{D}_1(z)u_{k-n+1}(t) + \bar{D}_0(z)u_{k-n}(t)$$

$$+ N_n(z)e_k(t + 1) + N_{n-1}(z)e_{k-1}(t + 1) + \cdots + N_1(z)e_{k-n+1}(t + 1) + N_0(z)e_{k-n}(t + 1)$$

which has the supervector expression

$$U_{k+1} = \bar{D}_nU_k + \bar{D}_{n-1}U_{k-1} + \cdots + \bar{D}_1U_{k-n+1} + \bar{D}_0U_{k-n}$$

$$+ N_nE_k + N_{n-1}E_{k-1} + \cdots + N_1E_{k-n+1} + N_0E_{k-n}$$

• 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 accommodated 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

$$\bar{D}_c(w) = I w^{n+1} - \bar{D}_{n-1}w^n - \cdots - \bar{D}_1 w - \bar{D}_0$$

$$N_c(w) = N_n w^n + N_{n-1}w^{n-1} + \cdots + N_1 w + N_0$$

• 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:

\[
G_{cl}(w) = H_p[I + C(w)H_p]^{-1}C(w)
\]

\[
= H_p[D_c(w) + N_c(w)H_p]^{-1}N_c(w)
\]

- 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.
Iteration-Varying Uncertainty

- 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.

- Example: 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

\[
    u_{k+1}(t) = u_{k-1}(t) + \gamma e_{k-1}(t + 1) \quad \text{with} \quad \gamma = 0.9 \Rightarrow C(w) = \frac{1}{(w^2-1)} L
\]
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 poles 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.
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.
- $C_{ILC}(w)$ denotes the ILC update law.
- $C_{CITE}(w)$ 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

<table>
<thead>
<tr>
<th>$Y_d$</th>
<th>$D$</th>
<th>$N$</th>
<th>$C$</th>
<th>$H$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_d(z)$</td>
<td>0</td>
<td>0</td>
<td>$\Gamma(w - 1)^{-1}$</td>
<td>$H_p$</td>
<td>Classical Arimoto algorithm</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>0</td>
<td>0</td>
<td>$\Gamma(w - 1)^{-1}$</td>
<td>$H_p(w)$</td>
<td>Owens’ multipass problem</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>$D(w)$</td>
<td>0</td>
<td>$C(w)$</td>
<td>$H_p$</td>
<td>General (higher-order) problem</td>
</tr>
<tr>
<td>$Y_d(w)$</td>
<td>$D(w)$</td>
<td>0</td>
<td>$C(w)$</td>
<td>$H_p$</td>
<td>General (higher-order) problem</td>
</tr>
<tr>
<td>$Y_d(w)$</td>
<td>$D(w)$</td>
<td>$N(w)$</td>
<td>$C(w)$</td>
<td>$H_p(w) + \Delta(w)$</td>
<td><strong>Most general problem</strong></td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>$D(z)$</td>
<td>0</td>
<td>$C(w)$</td>
<td>$H(w) + \Delta(w)$</td>
<td>Frequency-domain uncertainty</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>0</td>
<td>$w(t), v(t)$</td>
<td>$\Gamma(w - 1)^{-1}$</td>
<td>$H_p$</td>
<td>Least quadratic ILC</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>0</td>
<td>$w(t), v(t)$</td>
<td>$C(w)$</td>
<td>$H_p$</td>
<td>Stochastic ILC (general)</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>0</td>
<td>0</td>
<td>$\Gamma(w - 1)^{-1}$</td>
<td>$H^I$</td>
<td>Interval ILC</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>$D(z)$</td>
<td>$w(t), v(t)$</td>
<td>$C(w)$</td>
<td>$H(z) + \Delta H(z)$</td>
<td>Time-domain $H_\infty$ problem</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>$D(w)$</td>
<td>$w(k, t), v(k, t)$</td>
<td>$C(w)$</td>
<td>$H(w) + \Delta H(w)$</td>
<td>Iteration-domain $H_\infty$ ILC</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>0</td>
<td>$w(k, t), v(k, t)$</td>
<td>$\Gamma(k)$</td>
<td>$H_p + \Delta H(k)$</td>
<td>Iteration-varying uncertainty and control</td>
</tr>
<tr>
<td>$Y_d(z)$</td>
<td>0</td>
<td>$\tilde{H}$</td>
<td>$\Gamma$</td>
<td>$H_p$</td>
<td>Intermittent measurement problem</td>
</tr>
</tbody>
</table>

...
On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

Outline

• Introduction
  – Control System Design: Motivation for ILC
  – Iterative Learning Control: The Basic Idea
• On the History and Accomplishments of ILC
  – Early Work and Literature Survey
  – Selected Applications
  – Some ILC Algorithms
• An ILC Framework for the Past, Present, and Future
  – General Operator-Theoretic Framework
  – The “Supervector” Notation for Discrete-Time ILC
  – The $w$-Transform: “$z$-Operator” Along the Repetition Axis
  – ILC as a MIMO Control System
  – The Complete Framework: Iteration-Domain Variation
• Concluding Remarks: Future Research Vistas in ILC
  – Along the Same Lines: ILC for Spatial Processes (PDEs)
  – Something a Bit Different: Discrete Repetitive Processes
  – On the Far Horizon?: Iterative Learning and Extrapolation in Discrete Event Dynamic Systems

ANNIE2009 – Kevin L. Moore, Colorado School of Mines
Along the Same Lines – 1

• Thanks to the pioneering contributions of Professor Arimoto and the later researchers inspired by his work, ILC is now a technique that is
  – relatively-mature
  – inherently-“robust”
  – less model-based
  – tolerant-to-slight-nonlinearities

• Further, there is
  – Significant body of literature
  – Significant industrial applications

• So, … what’s next?
Along the Same Lines – 2

Following traditional lines, additional research remains:

- Transient and monotonic convergence issues are not fully studied
- Robustness still needs to be better understood
- Limits of performance
  - Repetitive control must obey the “waterbed effect.”
  - ILC may not have such a requirement in the time domain, due to the resetting operation, but ILC must obey the waterbed effect in the iteration domain.
  - Research remains to understand this issue
Along the Same Lines – 3

Following traditional lines, additional research remains:

• Sampled-data ILC has not been completely explored
  – Nonuniform sampling
  – Asynchronous sampling
  – Networked control ILC

• Joint time-frequency domain ILC techniques

• Extrapolation to new reference trajectories
  – How does learning one reference help with a second one
  – ILC with memory

• Nonlinear updating?:
  – Systematic theory of ILC for nonlinear systems is still open
Along the Same Lines – 4

“New” areas:

- ILC for fractional-order dynamic systems (polymer/piezo/silicon gel, etc.)

- ILC for large-scale, uncertain, spatial-temporal, interconnected systems will be increasingly important
  - Cooperative ILC in networks of agents

- ILC for Spatial and/or Temporal Processes:
  - ILC with vision feedback
  - Applications to PDE (partial differential equation) systems
ILC in Space and Time

- To date, most iterative or repetitive control has considered the repetition as being with respect to time

- Another approach is to consider repetition with respect to space

- Ultimately, the point becomes consideration of spatial-temporal processes

Two Examples:

- Vision-based ILC
- A diffusion process
Vision-Based, Spatial ILC: An Example

- Laser Pointer
- Target Path
- Camera
- Image Capture
- ILC Algorithm
- Motor Control
- (two computers)
On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

Gimbal Motion (k = 1)  
Gimbal Motion (k = 4)
ILC for a Diffusion Process

Center Pivot Irrigator Control

• Assume can bury a sensor at some prescribed depth at regular intervals
• Adjust water application rate based on the sensor readings between cycles
On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

ILC for a Diffusion Process (cont.)

The flow on the current pass

Variable Diffusion Coefficient

The maximum moisture measured during the previous pass

Plant

\[
\begin{aligned}
\frac{\partial v}{\partial t} &= -\gamma \frac{\partial v}{\partial x} \\
\frac{\partial c}{\partial x} + D \frac{\partial^2 c}{\partial x^2} &= \frac{\partial (vc)}{\partial x}
\end{aligned}
\]

The applied flow on the previous pass

ILC/RC-like update equation

\[
F(x, y, t) = \sum_{i=1}^{N} \left[ F(x, y, t - T_p) + \tau f(y_d - c(x, y, z_s, t')) \right] \delta_i(x, y, t)
\]

\[
f(y_d - c(x, y, z_s, t')) = \max_{t' \in [t - T_p, t]} y_d - c(x, y, z_s, t')
\]

ANNIE2009 – Kevin L. Moore, Colorado School of Mines
With no ILC:
- Areas with higher diffusion rate get less water

With ILC:
- Areas with varying diffusion rates get same water
On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm
Something a Bit Different: Discrete Repetitive Processes

• There are a number of repetitive processes where the results of the previous iteration affect the plant seen by the controller in the next iteration

• Material removal or application processes are prototypical

• Examples:
  – Diffusion process
  – Digging
  – Multi-pass welding
  – Soil compaction

Rather than ILC, these processes better fit a class of systems called **Repetitive Processes**
Digging: Shovel Automation

In each case, we might compute

\[ \text{Trajectory}_\text{(k+1)} = f(\text{Profile}(k+1)) + \text{Trajectory}(k) \]
On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

Mutli-Pass Welding

- Wirefeed speed, Welding voltage, CT
- Forward motion
- Weave motion
- Second Pass
- First Pass
Soil Compaction
Modeling Repetitive Processes

Normal Discrete-Time Linear Model (no repetition)

\[ x(p + 1) = Ax(p) + Bu(p) \]

\[ y(p) = Cx(p) + Du(p) \]

\[ 0 \leq p \leq \alpha \]

Normal Discrete-Time Linear Model (with repetition)

\[ x_k(p + 1) = Ax_k(p) + Bu_k(p) \]

\[ y_k(p) = Cx_k(p) + Du_k(p) \]

\[ 0 \leq p \leq \alpha \]

\[ k \text{ is trial or iteration} \]

\[ p \text{ is time (or space)} \]
Iterative Learning Control

Normal Discrete-Time ILC

\[ x_k(p + 1) = Ax_k(p) + Bu_k(p) \]
\[ y_k(p) = Cx_k(p) + Du_k(p) \quad 0 \leq p \leq \alpha \]

ILC adds this control law to normal linear model:
- Modifies control not plant
- Adds memory to the control law
Discrete Repetitive Processes

Discrete Repetitive Processes (DRP)

\[ x_{k+1}(p + 1) = Ax_{k+1}(p) + Bu_{k+1}(p) \]
\[ y_{k+1}(p) =Cx_{k+1}(p) + Du_{k+1}(p) \]

Boundary conditions:
\[ x_{k+1}(0) = d_{k+1} \]

Control:
\[ u_{k+1}(p) = f(y_{d}(p) - y_{k+1}(p)) \]

DRP adds this to normal linear model
- Modifies plant, not control
- Adds memory to the plant
Extending the DRP Control

Discrete Repetitive Processes (DRP)

\[ x_{k+1}(p + 1) = Ax_{k+1}(p) + Bu_{k+1}(p) + B_0y_k(p) \]
\[ y_{k+1}(p) = Cx_{k+1}(p) + Du_{k+1}(p) + D_0y_k(p) \quad 0 \leq p \leq \alpha \]

Boundary conditions:
\[ x_{k+1}(0) = d_{k+1} \]

Control:
\[ u_{k+1}(p) = f_1(y_d(p) - y_{k+1}(p)) \]

Add previous cycle control terms, like ILC.
Extensions to These Models

- DRP models do not completely capture the spatial-temporal repetitive process applications described earlier
- Example: soil compaction:

Effect of force at location $x_0$ is felt in all directions around $x_0$ (in 3 D)
Extending the DRP Model

Discrete Repetitive Processes (DRP)

\[ x_{k+1}(p+1) = Ax_{k+1}(p) + Bu_{k+1}(p) + B_1y_k(p) \]

\[ y_{k+1}(p) = Cx_{k+1}(p) + Du_{k+1}(p) + D_1y_k(p) \]

Boundary conditions:

\[ x_{k+1}(0) = d_{k+1} \]

Control:

\[ u_{k+1}(p) = f(y_d(p) - y_{k+1}(p)) \]

Need to add these terms to the DRP model, if we equate time with space.

0 \leq p \leq \alpha
Various Issues to Consider

- Investigate convergence conditions once a suitable modelling framework is defined
- Study the connection to standard multi-dimensional processes
  - Can view the spatial-temporal repetitive process as an nD repetitive process
- Consider repetitive processes that have a variable pass length in time but fixed in space (account for actuator motion)
- Understand how these models are special cases of regularized PDEs and then see if we can solve them by formulating and solving the ILC problem for PDEs
On the Far Horizon?: Iterative Learning and Extrapolation in Discrete Event Dynamic Systems (toward Intelligent, Autonomous systems)

From Control ... (PID, Bode) ... to Intelligent Control ... (Fuzzy, Neural, GA, etc.) ... to Intelligent Behavior ... (Hybrid, etc.) ... to Cooperative Autonomy ... (Agent-based, etc.)
Enablers for Intelligent, Autonomous Systems

We consider two key aspects of autonomy:

- **Inherent physical capabilities**
  - Mechanisms for mobility and manipulation
  - Sensors for perception
    - Proprioceptive
    - External
  - Computational power

- **Intelligent control to exploit these capabilities**
  - Machine-level control
  - Perception algorithms
  - Reasoning and decision-making
  - Learning

Much progress has been made in this area.
Comments on Intelligent Control -1

- “Intelligent control” is a well-used phrase
- In IEEE CSS society, intelligent control has come to be called “computational intelligence” and is often associated with
  - Fuzzy logic
  - Neural nets
  - GAs
  - Adaptive and learning control
- It was in the context of adaptive and learning control that I began working on my main research topic:
  - Iterative learning control, or ILC
What is “Intelligent Control?”

  
  …. has the ability to **act appropriately in an uncertain environment**, where an appropriate action is that which increases the probability of success and success is the achievement of behavioral subgoals that support the system’s ultimate goal
  
  …. envisioned as **emulating human mental faculties** such as adaptation and learning, planning under large uncertainty, coping with large amounts of data, etc.
  
  …. aims to attain **higher degrees of autonomy** and even setting control goals rather than stressing the intelligent methodology that achieves those goals.

- Another definition: intelligent control
  
  ... is the use of a general purpose control system which **learns over time** how to **achieve goals** (or optimize) in **complex environments** that are noisy, nonlinear and possibly unknown or uncertain
Comments on Intelligent Control -2

- If “success = use,” then we can conclude that some aspects of the promise of intelligent control have been achieved:
  - ILC has had a fairly successful history:
    - Well-established literature of analysis and design techniques
    - Well-established practice as evidenced by several patents related to commercial products
    - Even a PaperPlaza keyword for ACC/CDC/ISIC/CCA
  - Artificial neural nets are ubiquitous (at least Feedforward ANNs)
  - Any/every undergrad with the Matlab Fuzzy toolbox does FLC
- But: are techniques like NN, FLC, GA, or ILC, built into suitable architectures, which have the word “learning” in their title, really intelligent?
Comments on Intelligent Control -3

• Likewise, there have been a large number of ideas on architectures:
  – Subsumption (Brooks).
    • Input-output relations to solve small problem; prioritization
  – Behavior-based hierarchical (Arkin)
    • Motor schemas
  – Behavior-based reinforcement learning (Barto/Sutton/Anderson)
    • Output of a difference engine measuring state-goal mismatch and taking action to minimize that mismatch (Minsky)
  – Deliberative/Reactive
    • AI-based planning approach
  – 4D/RCS (Albus)
Whither Intelligent Control? -1

- Are we there yet?
- In my opinion: No
- Most architectures are “best guess” engineering approximations to current state of knowledge about biological function and its organization
- Most NN or FLC controllers are better described as biologically-inspired computational elements
  - Primarily compute I/O maps (nonlinear) for use in feedback control systems
  - Do some pattern recognition tasks
  - Self-organize to achieve a functional property
- Most “learning” algorithms are really better called “adaptation” algorithms doing parameter or function update
## Control → Intelligent Control

<table>
<thead>
<tr>
<th>Fixed designs</th>
<th>Flexible designs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Feedback</td>
<td>Adaptive Learning Control</td>
</tr>
<tr>
<td>Dynamic Compensation</td>
<td>Change parameter $p$</td>
</tr>
<tr>
<td>$u(s) = Ke(s)$</td>
<td>Change controller $K$</td>
</tr>
</tbody>
</table>

All that is really happening is the controller (or parameter) is being adapted → Increasing uncertainty → Increasing use of performance feedback → Increasing use of past information and experience → Precision loses intelligence / Intelligence loses precision

(Due to George Saridis)
### Adaptation versus Learning

(adaption and learning are fundamentally different)

<table>
<thead>
<tr>
<th>Adaptation</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactive: maintain desired behavior (local optimization)</td>
<td>Constructional: synthesize desired behavior (global optimization)</td>
</tr>
<tr>
<td>Temporal emphasis</td>
<td>Spatial emphasis</td>
</tr>
<tr>
<td>No memory $\rightarrow$ no anticipation</td>
<td>Memory $\rightarrow$ anticipation</td>
</tr>
<tr>
<td>Fast dynamics</td>
<td>Slow dynamics</td>
</tr>
<tr>
<td>Known structure and slowly time-varying behavior</td>
<td>Structural uncertainty and nonlinear dependencies</td>
</tr>
</tbody>
</table>

(Due to ??)
Whither Intelligent Control? -2

• Repeat: for most “learning controllers” to date in the intelligent control field:
  – “Learning” is usually only parameter or functional (or model-free) adaptation
• (Iterative updating is not learning; parameter adaptation is not learning)
• True learning requires adaptation at the meta-level
• What is needed are better understandings of
  – Purpose of intelligence (goals)
  – Components of intelligence (memory, learning)
  – Organization of intelligence (models, language, architecture)
Whither Intelligent Control? -2

• To reinforce this point, consider (from “Future Directions in Control in an Information-Rich World,” Murray, Åström, Boyd, Brockett, Stein, IEEE Control Systems Magazine, April 2003):

“The role of logic and decision-making in control systems is becoming and increasingly large part of modern control systems … includes … higher levels of abstract reasoning using high-level languages … effective frameworks for analyzing and designing systems of this form have not yet been fully developed”

• This leads us to consider the proper modeling and organization of the components of intelligence

• Suggest that a language-based formalism is the way to approach this
Example: ODIS FindCar() Script
This example was achieved using a parameterized, script-based language

- **findCar() script**
  
  If (car) fit bumper and move in
  
  fire sonar at rear of stall
  
  if there is something in the stall
  
  fire sonar at front half of stall
  
  fit bumper_line
  
  move to $\cap$ of bumper_line with c.l. of stall
  
  fit tire_ol
  
  ...
  
  else go to next stall

- This script was implemented in a suitable architecture
Hybrid Systems Approach

- The mobile robot behavior generator can be interpreted as a hybrid system (discrete event (logic) dynamics plus continuous dynamics)

  - In this interpretation commands and events are symbols in an alphabet associated with a (regular) language
  - This formalism can be used for synthesis of scripts
  - The frontier for ILC is to develop ways to “learn” scripts (e.g., synthesize scripts of actions based on results from previous scripts)
On the History, Accomplishments, and Future of the Iterative Learning Control Paradigm

Outline

• Introduction
  – Control System Design: Motivation for ILC
  – Iterative Learning Control: The Basic Idea
• On the History and Accomplishments of ILC
  – Early Work and Literature Survey
  – Selected Applications
  – Some ILC Algorithms
• An ILC Framework for the Past, Present, and Future
  – General Operator-Theoretic Framework
  – The “Supervector” Notation for Discrete-Time ILC
  – The w-Transform: “z-Operator” Along the Repetition Axis
  – ILC as a MIMO Control System
  – The Complete Framework: Iteration-Domain Variation
• Concluding Remarks: Future Research Vistas in ILC
  – Along the Same Lines: ILC for Spatial Processes (PDEs)
  – Something a Bit Different: Discrete Repetitive Processes
  – On the Far Horizon?: Iterative Learning and Extrapolation in Discrete Event Dynamic Systems
Acknowledgments

Professor Suguru Arimoto
Ritsumeikan University

Professor YangQuan Chen
Utah State University (USU)
Center for Self-Organizing and Intelligent Systems (CSOIS)

Professor Hyo-Sung Ahn
Gwangju Institute of Technology

Many Other Collaborators:
Eric Rogers, University of Southampton
David Owens, Sheffield University
Jari Hatonen, Borealis Group.
Mikael Norloff, ABB and Linkoping U.
Jin-Xin Xu, National University of Singapore
Dan Wang, Nanyang Tecnological Institute
Richard Longman, Columbia University
Minh Phan, Dartmouth University
... and many others!
To Read More


