A Retrospective on Adaptive Dynamic Programming for Control

IJCNN’2009
Atlanta, GA    June 14, 2009

George G. Lendaris
NW Computational Intelligence Laboratory
Portland State University, Portland, OR

Partial support by US National Science Foundation ECS-0301022
A PARSING of the HISTORY of the CONTROLS FIELD

Phase 1: Design based on intuition and invention.

Phase 2: Design based on mathematical tools.

Phase 3: Design for accommodating context variations

Phase 4: Design for EXPERIENCE-BASED PROCESSES including autonomous Context Discernment and Model Selection
Phase 1: Design based on intuition and invention.

a) Control devices date back to antiquity.

b) Flyball governor invented by James Watt in 1788.

Characterize this Phase as follows:

Design of control devices based largely on intuition and inventive genius, with little or no support from mathematically based tools, and no explicit notion of context.
Phase 2: Design based on mathematical tools.

Mathematics has played a fundamental role in the development of the controls field as it is understood today.

I mark beginning of this phase with Maxwell’s use of differential equations to analyze the flyball governor’s dynamics (ca. 1870).

Progression: Fourier transforms, Laplace transforms, state space methods, stochastic methods, Hilbert space methods, and more recently, algebraic and geometric topological methods.
Phase 2 (cont.):

Characterize this Phase as follows:

Contains design methods wherein the resulting controller is to be placed in service with *no associated mechanism for modifying its design* in response to changes in either the plant or its environment [context].

This phase thus includes at least the following design methods (using well known labels from the controls literature):

Classical Control, Modern Control, Optimal Control, Stochastic Control and Robust Control.

[Design is done off-line, prior to installation.]

[Accommodate changes via “margins” rather than adjustments.]
Phase 3: Design for accommodating context variations

In some applications, context changes so much during operation that fixed controller designs resulting from Phase 2 methods are not sufficient.

Approaches emerged for accommodating such contextual variations via on-line instantiation of different controller designs based on these variations.

A. Partitioning Methods
B. Adaptive Control
C. Learning Control
Phase 3 (cont.):

Phase 3 is here characterized as a design approach that:

1) Yields controllers for which selected on-line adjustment of parameter values is allowed.
2) Accommodates a modicum of variations in context variables.
3) Mechanism for performing the ‘accommodations’ is distinct from that to be defined for the new Phase 4.
Phase 3 (cont.):

A. Partitioning Methods

1. Partition a nonlinear operating region into approximately linear regions [each becoming a different control context] and develop a linear controller appropriate to each region.

2. Various methods have different means of “knowing” which context is the current one.

3. Once the specific current context is known, a previously designated controller or controller design process is then instantiated.
Phase 3 (cont.):

A. Partitioning Methods (cont.)

Partitioning methods appear in a variety of technology sectors: control theory, artificial intelligence, neural networks, Fuzzy logic, statistics, etc.

Associated methods appear under a variety of labels: multiple models, piecewise models, mixture of experts, Fuzzy models, local regression, etc.

These methods differ from methods to appear in the new Phase 4 is that in Phase 3 methods, adjustments to controller design are pre-specified by the human designers.
Phase 3 (cont.):

B. Adaptive Control

Allows multiple policies to be available for performing the desired control.

Traditionally grouped into two separate classes:

- Model-Reference Adaptive Controller (MRAC), and
- Self-Tuning Controller (STC).

- In both classes, have “outer loop” that performs adjustments to controller parameters (also called tuning).

[Gain Scheduling is simplest form of such control.]
Phase 3 (cont.):

B. Adaptive Control (cont.):

MRAC:
  • Specifications given in terms of a reference model (indicates how the plant output should respond ideally to the command signal).
  • Controller parameters adjusted to make the plant output close to the model outputs.

STC:
  • Specifications given in terms of a performance index.
  • There is an on-line identifier to estimate current values of plant parameters; these estimates are used by online “design block” to design a controller to meet the performance index.
Phase 3 (cont.):

B. Adaptive Control (cont.):

Recap:

Basis is a parameterized controller model (of pre-designed structure), and the adaptation refers to changes made in its parameter values.

Adjustments are made to accommodate performance errors resulting from inaccurate plant models, and/or changes in the plant that is being modeled (i.e., changes in context).
Phase 3 (cont.):

B. Adaptive Control (cont.):

Notes:

• Structure of the control policy is pre-designed (off-line) for the engineering task at hand.

• Engineer’s job is to implement a principled method for on-line **tuning** of the control policy (via incremental adjustments to controller parameter values), based on observations.

• A feature that distinguishes Adaptive Control methods from those to appear in Phase 4 is that controller **design tweaking** (“tuning”) occurs on-line in Adaptive Control, in contrast to on-line **selection from a pool** of previously designed controllers (with possible refinement) to be described for Phase 4.
Phase 3 (cont.):  

B. Adaptive Control (cont.):  

In Learning Control,  

• Less is assumed known about the plant.  
  • Controller model (being adjusted) is of more general form than used in Adaptive Control.  

Engineer’s job:  

• Specify a parameterized controller model, with less a priori information embedded in it than typical for Adaptive Control case.  
• Provide the learning algorithm used to iteratively arrive at an optimal policy, based on a sequence of state observations and performance evaluations.  

Policy resulting from such Learning Control is typically capable of covering larger portion of state space than via previous methods.
### Comparison of Adaptive vs. Learning Control

<table>
<thead>
<tr>
<th>Adaptive vs. Learning Control</th>
<th>Candidate Pool of Policies</th>
<th>Selection Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adaptive Control.</strong> Much a priori information about plant and environment is available and used.</td>
<td>Carefully selected set of policies relevant to the engineering task (often in terms of a parameterized model).</td>
<td>Tailored (by the engineer) to the pool of policies. Switches between policies during on-line operation, albeit some times slowly.</td>
</tr>
<tr>
<td><strong>Learning Control</strong> (of ADP type). Little or no a priori information about plant and/or environment is available or used.</td>
<td>Broadly selected set of policies for the task (also parameterized model, but with less <em>a priori</em> knowledge embedded in its structure, e.g. all controllers that can be instantiated by an MLP).</td>
<td>Machine based, general and often slow. Historically, selects a single policy during offline training, which is then used for online operation, with no adaptation. More recently, preliminary design is done off-line, and on-line adaptation is permitted.</td>
</tr>
</tbody>
</table>
Phase 4: [New] Design for EXPERIENCE-BASED PROCESSES, including autonomous Context Discernment and Model Selection

Design approach that develops an (autonomous) Agent to fill the *experienced designer* role in the following metaphor:

A plant, environment, and control objective are provided to the designer, who is to design/select a controller. If the designer is experienced and has “seen” the problem before, after gathering context data, he/she pulls the appropriate design out of the archives and applies it to the current situation – perhaps with a little tailoring. The more experience the control designer possesses, the process goes faster and with better results [efficient & effective].
Phase 4 (cont.):

Stipulated Requirements for Phase 4:

Agent has the ability to:

a) use experience for model selection (plant or controller); and

b) do so effectively and efficiently.
Phase 4 (cont.):

Fundamental aspects to consider:

a) context, b) discerning current context,

c) selecting appropriate model from experience repository for the discerned context, and

d) doing the latter two in an effective and efficient manner.
Phase 4 (cont.):

To accomplish Experience-Based Control (EBC), need to:

1) Accumulate and/or select a set of “relevant” policies for an engineering task (as is done by the engineer for Adaptive Control) --> Experience Repository.

2) Create a representation schema that facilitates accessing the policies.

3) Create an algorithm that effectively and efficiently switches between these policies (adapts) during online operation of the system, in response to changes in its context.

The process may entail a Higher Level Learning Algorithm (HLLA):

– In sense that the learning is occurring at a higher level of operation than is normally performed in Learning Control.

– “Higher Level” refers to idea that algorithm is to evolve a strategy for selecting an optimal controller from an available pool in an optimal way,

– in contrast to (the usual application of) adjusting “tweaking” the existing controller’s design.

NOTE: The two uses of the word ‘optimal’ refer to two distinctly different Criterion Functions: the first is related to the control task, and the second is related to the process of selecting the corresponding optimal controller.
Phase 4 (cont.): Approach 1: Direct access to experience in NN repository.

i) Embed multiple memories (“experience”) in NN, and index them via Context Variable(s) input. [“Contextually Aware Controller”]

ii) This entails predefining mechanism(s) for context discernment and employ as “Context Variable” (CV) input to NN controller during ADP training.

iii) During operation, the NN has learned to employ the CV input as “index” to the embedded distinct controller designs for different contexts.
Standard Use of ADP

Single Controller in NN Designed/Trained via ADP

For Phase 4 Approach 1: NN as Repository, with Context Variable (CV) as Selector

[“Contextually Aware Controller”]

NN Designed via ADP with array of CV values results in multiple embedded controllers.

[In operation, CV serves as SELECTOR for the different Controllers.]
Phase 4 (cont.): Approach 2:

Employ ADP methods to develop means for performing selection of controller from a repository instantiated differently than in Approach 1.

Employ ADP methods to implement Context Discernment process (includes system identification for control application).

An ADP method called HLLA is employed for both.
KEY IDEA of HLLA:

- Re-purpose the Reinforcement Learning method (to a “higher level”) such that
- Instead of using it to design an optimal controller for a given task,
- An already achieved collection of such solutions for a variety of related contexts is provided
  – (as an experience repository), and
- HLLA creates a strategy for optimally selecting a solution from the repository.
Phase 4 controller design via ADP – Approach 2 (HLLA)

[Repeat last component from previous slide:

HLLA creates a strategy for optimally selecting a solution from the repository.]

Research Stage 1: System Identification Task

Here, the Repository comprises collection of plant models (in contrast to controller models), and task is to optimally select a model that matches plant being identified.
HLLA Stage 1, Example 1: Pole-Cart Problem (Revisited)

Assume:

1) A controller for nominal Pole-Cart is in operation.
2) Pole-Cart attributes change suddenly.
3) For controller to “adapt”, needs to find present condition of the Pole-Cart. [Context Discernment; System Identification.]

Method:

a) Craft a “repository” of various versions of the Pole-Cart plant.

b) Develop HLLA process to optimally select (wrt quality of controller, and speed of access) model from the repository that matches current plant condition.
HILLA System Identification (SysID) for Pole-Cart, cont.

Approach taken:

a) Employed equations of motion of Pole-Cart plant as the “repository”. Changes in plant accomplished via changes in parameter values of the equations.

b) Form of equations employed had mass and length of pole among its parameters. This enabled creation of “plant manifold” [see appendix] and defining a coordinate space in terms of the parameters.

[Description of overall HILLA process → next slide:]
HLLA Overview:

"Controller" role for ADP

CDN (Context Discerning Network)

$\Delta C_D(t)$

$C_D(t)$

$R_A(t)$

$u(t)$

"Plant" role for ADP

Critic

$\lambda(t)$

$U(t) = (R_A(t+1) - R_D(t+1))^2$

$D(t) = R_A(t+1) - R_D(t+1)$

Used To Train CDN And Critic

Used To Train Critic
HLLA SysID for **Pole-Cart** – Experimental Results:

**TOP:** Context Discernment in response to change in plant parameter values (context change) at every 50th iteration.

**BOTTOM:** Errors between values for pole-cart system state variable and for models selected during discernment process.
• HLLA Stage 1, Example 2: Neural Network as Plant

• Approach taken:
  – Crafted a neural network of specified structure and element type (MLP) to be “repository”. Changes in plant accomplished via changes in selected weight values of NN.
  – Form of equations associated with MLP have weights as “parameters”. This enabled creation of “plant manifold” [see appendix] and defining a coordinate space in terms of the weights/parameters.
  – [Description of overall HLLA process ➔ next slide:]
• Neural Network as Plant: Experiment details

• NN parameters (weights and biases) are continuous.
• Connection weights are held fixed, bias weights allowed to vary.
• Each setting of the bias weights defines a unique NN.
• Generate a set of such NNs by varying the bias weight values.
• Let the biases be the coordinates $\rightarrow$ neural manifold (see Appendix)
• Procedure is equivalent to one used for Pole Cart (cf. HLLA overview slide).
• Context Discerner’s task: Determine NN in repository that corresponds to the specific one currently instantiated in the System box.
• Included NNs with different ratios of fixed to variable parameters, with good success involving some 40 parameters, and ratios of about five to one.
• Observe that at each change of NN in the System box, the context discerner successfully noted change and rapidly selected the corresponding NN from the repository.
• Error plot demonstrates a spike in error value when the change occurs, followed by a quick recovery – 10 or fewer observations of context used.
HLLA SysID for NN as Plant – Experimental Results:

TOP: Context Discernment in response to change in NN plant parameter values (context change) at every 100th iteration.

BOTTOM: Errors between values for pole-cart system state variable and for models selected during discernment process.
Proposed model of Experience-Based Control Configuration

Starting Condition:

- CONTROLLER
- PLANT

Criterion Function Assessor (CFA)

Context Monitoring

Perform Controller SELECTION (EB)

EB UPDATED PLANT MODEL

Perform (EB) SID

EB-UPDATED CONTROLLER MODEL

EB-UPDATED PLANT MODEL

Criterion Function Assessor (CFA)

All OK

Off Nominal

Off Nominal

All OK

Install Updated Controller Design

Run Simulation
OVERALL CONCLUSION
ABOUT EB APPROACH

I conjecture that the proposed *experience-based* approach will usher in a whole new phase of development of the decision and controls fields – making a significant stride toward the achievement of more human-like decision and control.

Also conjecture that the *context discernment* concepts plus the *manifolds representation* will provide a basis for constructing learning agents capable of long term *rapidly accessible memory*. If so, this could pave the way for scaling neural systems to brain-like capabilities.
Papers in this session:

- A Retrospective on Adaptive Dynamic Programming for Control
  George G. Lendaris
- Generalized Policy Iteration for Continuous-Time Systems
  Draguna Vrabie and Frank Lewis
- Adaptive Dynamic Programming-based Optimal Control of Unknown Affine Nonlinear Discrete-time Systems
  Travis Dierks, Balaje Thumati and S. Jagannathan
- SNAC Convergence and Use in Adaptive Autopilot Design
  Songjie Chen, Yang Yang, Sivasubramanya Balakrishnan, Nhan Nguyen and Kalmanje KrishnaKumar
  Derong Liu and Ning Jin
- Online Actor Critic Algorithm to Solve The Continuous-Time Infinite Horizon Optimal Control Problem
  Kyriakos Vamvoudakis and Frank Lewis
Basic Control Design Problem

Controller designer needs following:

- Problem domain specifications
- Design objectives / Criteria for “success”
- All available *a priori* information about Plant and Environment
In this tutorial, we focus on the Adaptive Critic Method (ACM)

ACM is a methodology for designing an (approximately) optimal controller for a given plant according to a stated criterion, via a learning process.

ACM may be implemented using two neural networks (also Fuzzy systems):

---> one in role of controller, and

---> one in role of critic.
Background: Simple 1-Hidden Layer, Feedforward

Neural Network [ to implement $y = f(x)$ ]

Can obtain partial derivatives via Dual of trained NN
Dual of Feedforward NN:

Backprop Training Rule: \[
\Delta w_{ji} = \beta o_i \delta_j
\]

Dual can also be used to evaluate Jacobian [Verbal Description.]
Overview of Adaptive Critic method

**User provides** the

Design objectives / Criteria for “success” through a Utility Function, \( U(t) \) (local cost).

Then, a new utility function is defined (Bellman Eqn.),

\[
J(t) = \sum_{k=0}^{\infty} \gamma^k U(t+k)
\]

which is to be minimized \[\sim\] Dynamic Programming.

[We note: \( J(t) = U(t) + \gamma J(t+1) \)] \( \leftarrow \) Bellman Recursion
Family of Adaptive Critic Methods

The critic approximates either $J(t)$ or the gradient of $J(t)$ wrt state vector $R(t)$ \[ \nabla J(R(t)) \]

Two members of this “family”:

**Heuristic Dynamic Programming (HDP)**

- **Critic** approximates $J(t)$
  (cf. “Q Learning”)

**Dual Heuristic Programming (DHP)**

- **Critic** approximates $\nabla J(R(t)) \equiv \lambda(t)$

[Today, focus on DHP]
May describe **DHP** training process via two primary feedback loops:
DHP’s two training loops:

1. The **controller training loop**. A Supervised Learning process in which the controller is trained to minimize the performance measure $J(t)$ of the control problem, based on data from the critic [called $\lambda(t)$].

2. The **critic training loop**. The process wherein the critic neural network learns to approximate the derivatives of the performance measure $J(t)$, which are used in the **controller training loop**.
To develop feel for the weight update rule, consider a partial block diagram and a little math (discrete time):

Desire a training “Delta Rule” for $w_{ij}$ to minimize cost-to-go $J(t)$.

Obtain this via $\frac{\partial J(t)}{\partial w_{ij}(t)}$ and the chain rule of differentiation.
Basic concept of CHAIN RULE of differentiation:

\[
\frac{\partial J(t+1)}{\partial w_{ij}} = \frac{\partial J(t+1)}{\partial R(t+1)} \frac{\partial R(t+1)}{\partial u(t)} \frac{\partial u(t)}{\partial w_{ij}}
\]

(Ordered derivatives.)
The weights in controller NN are updated with objective of minimizing $J(t)$:

$$\Delta w_{i,j}(t) = -l\text{coef} \cdot \frac{\partial J(t)}{\partial w_{ij}(t)}$$  \hspace{1cm} (1)$$

where

$$\frac{\partial J(t)}{\partial w_{ij}(t)} = \sum_{k=1}^{a} \frac{\partial J(t)}{\partial u_k(t)} \cdot \frac{\partial u_k(t)}{\partial w_{ij}}$$  \hspace{1cm} (2)$$

and

$$\frac{\partial J(t)}{\partial u_k(t)} = \frac{\partial U(t)}{\partial u_k(t)} + \frac{\partial J(t + 1)}{\partial u_k(t)}$$  \hspace{1cm} (3)$$

and

$$\frac{\partial J(t + 1)}{\partial u_k(t)} = \sum_{s=1}^{n} \frac{\partial J(t + 1)}{\partial R_s(t + 1)} \cdot \frac{\partial R_s(t + 1)}{\partial u_k(t)}$$  \hspace{1cm} (4)$$

Call this term $\lambda_s(t + 1)$ (to be output of the critic)
It follows that **Controller training** is based on:

\[
\frac{\partial J(t)}{\partial u_k(t)} = \frac{\partial U(t)}{\partial u_k(t)} + \sum_{s=1}^{n} \frac{\partial J(t+1)}{\partial R_s(t+1)} \frac{\partial R_s(t+1)}{\partial u_k(t)}
\]

(5)

**Via CRITIC**

**Via Plant Model**

Similarly, **Critic training** is based on:

\[
\frac{\partial J(t)}{\partial R_s(t)} = \frac{dU(t)}{dR_s(t)} + \sum_{k=1}^{n} \frac{\partial J(t+1)}{\partial R_k(t+1)} \left( \frac{\partial R_k(t+1)}{\partial R_s(t)} \frac{\partial R_s(t)}{\partial u_m(t)} + \sum_{m} \frac{\partial R_k(t+1)}{\partial u_m(t)} \frac{\partial u_m(t)}{\partial R_s(t)} \right)
\]

**Via Plant Model**

**Via Controller**

[Bellman Recursion & Chain Rule used in above.]

Plant model is needed to calculate partial derivatives for DHP ...
Components of DHP type Adaptive Critic

[Dark boxes: analytic expressions; medium boxes: critical calculations; clear boxes: neural networks.]
is approximated by the critic, in response to the input $R(t+1)$.

\[ \frac{\partial R(t + 1)}{\partial u(t)} \] can be calculated from analytical equations of the plant, if they are available, or by backpropagation through a third neural net that has been previously trained to copy the plant.
The value for $\lambda_s^o(t)$ [bottom medium box in system diagram] is calculated via the following equation:

$$\lambda_s^o(t) = \frac{\partial U(t)}{\partial R_s(t)} + \sum_{j=1}^{a} \left( \frac{\partial U(t)}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial R_s(t)} \right) + \sum_{j=1}^{a} \left( \frac{\partial J(t+1)}{\partial R_k(t+1)} \frac{\partial R_k(t+1)}{\partial R_s(t)} \right)$$

$$+ \sum_{k=1}^{n} \left\{ \sum_{j=1}^{a} \left( \frac{\partial J(t+1)}{\partial R_k(t+1)} \frac{\partial R_k(t+1)}{\partial u_j(t)} \right) \right\}$$

[See following slide for paraphrased version.]
Paraphrased version of equation for calculating Target Value for λ_s^o (t) [bottom medium box in system diagram]
is the following:

\[
\lambda_s^o (t) = \sim \text{Utility} + \sum_{j=1}^{a} (\sim \text{Utility} \cdot \sim \text{Action}) + \sum_{j=1}^{a} (\sim \text{Critic}(t+1) \cdot \sim \text{Plant}) \\
+ \sum_{k=1}^{n} \left\{ \sum_{j=1}^{a} (\sim \text{Critic}(t+1) \cdot \sim \text{Plant} \cdot \sim \text{Action}) \right\}
\]
Strategies for training the two loops.
• **S1:** Simultaneous (Classical Strategy)
• **S2:** Flip-Flop Strategy
• **S4:** Shadow Critic Strategy
  - Introduce a copy of the `criticNN` in the lower loop.
  - Run both loops simultaneously.
    - **Lower loop:** train the copy for an epoch; upload weight values from copy (called `shadow` critic) into the active `criticNN`; repeat.
• **S5:** Shadow Controller Strategy
  - Introduce a copy of the `actionNN` in the upper loop.
  - Run both loops simultaneously.
    - **Upper loop,** train the copy for an epoch; upload weight values from copy (called `shadow` controller) into the active `actionNN`, repeat.
S6: Double Shadow Strategy

Make use of the NN copies in both training loops.

Run both loops simultaneously.

Both loops: train the copies for an epoch; upload weight values from the copies into their respective active NNs.

Key ADP Process Parameters

- Specification of:
  - state variables
  - size of NN structure, connection pattern, and type of activation functions
  - learn rate of weight update rules
  - discount factor (gamma of Bellman Equation)
  - scaling of variable values (unit hyper-sphere/cube)
  - “lesson plan” for training
  - training strategy
  - epoch size
Three Examples of employing DHP

1) Pole-Cart Benchmark Problem
2) Autonomous Two-Axle Terrestrial Vehicle
3) Aircraft Controller
Example 1 (for DHP):

Pole-Cart benchmark problem
Pole-Cart Benchmark Problem

For this example, we defined

\[ U(t) = -0.25 \dot{\theta}^2 - 0.25 x^2 \]
\[ R_6(t + 1) = \frac{g \sin \theta + \cos \theta}{m_c + m} \left[ -F - ml \dot{\theta}^2 \sin \theta + \mu_c \text{sgn}(\dot{x}) \right] - \frac{\mu_p \dot{\theta}}{ml} \]

\[ R_3(t + 1) = \frac{F + ml \left[ \dot{\theta}^2 \sin \theta - \dot{\theta} \cos \theta \right]}{m_c + m} - \mu_c \text{sgn}(\dot{x}) \]

\[ R_1(t + 1) = R_1(t) + \tau \cdot R_2(t); \quad R_2(t + 1) = R_2(t) + \tau \cdot R_3(t) \]

\[ R_4(t + 1) = R_4(t) + \tau \cdot R_5(t); \quad R_5(t + 1) = R_5(t) + \tau \cdot R_6(t) \]
Experimental Procedure ("lesson plan"): 

A. Train 3 passes through sequence (5, -10, 20, -5, -20, 10) [degrees from vertical]. Train 30 sec. on each angle.

B. Accumulate absolute values of U: C(1), C(2), C(3).

C. Perform TEST pass through train sequence (30 sec. each angle). Accumulate U values: C(4).

D. Perform GENERALIZE pass through sequence (-23, -18, -8, 3, 13, 23) [degrees from vertical] and accumulate U values: C(5).

E. Perform GENERALIZE pass through sequence (-38, -33, 23, 38) [degrees from vertical] and accumulate U values: C(6).
Progress of training process under each of the strategies.

[Pole angles during the first 80 sec. of training.]
[Note how fast Strategies 4a & 4b learn.]
Controller Responses to Disturbances

Pole angles during part of test sequence.

[Markings below and parallel to axis are plotting artifacts.]
Example 2 (for DHP):

Autonomous Two-Axle Terrestrial Vehicle
Autonomous Vehicle Example
(Two-axle terrestrial vehicle)

Objective: Demonstrate design via DHP ("from scratch") of a steering controller which effects a Change of Lanes (on a straight highway).

Controller Task: Specify sequence of steering angles to effect accelerations needed to change orientations of vehicle velocity vector.
Design Scenarios -- general:

Maintain approximately constant forward velocity during lane change.
Control angular velocity of wheel.
Thus, Controller needs 2 outputs:

1.) Steering angle
2.) Wheel rotation velocity
Major unknown for controller:

Coefficient of Friction (cof) between tire/road at any point in time.

It can change abruptly (based on road conditions [CONTEXT]).

Requires robustness and/or fast on-line adaptation.
Tire side-force/side-slip-angle as a function of COF
Equations used:

\[
\begin{bmatrix}
  m\nu(\beta + p) \\
  m\dot{\nu} \\
  mlf l_b \phi
\end{bmatrix} =
\begin{bmatrix}
  -\sin(\beta) & \cos(\beta) & 0 \\
  \cos(\beta) & \sin(\beta) & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  f_x \\
  f_y \\
  m_z
\end{bmatrix}
\]

\[
\begin{bmatrix}
  f_x \\
  f_y \\
  m_z
\end{bmatrix} =
\begin{bmatrix}
  -\sin(\delta_f) & 0 & 1 \\
  \cos(\delta_f) & 1 & 0 \\
  l_f \cos(\delta_f) & -l_b & 0
\end{bmatrix}
\begin{bmatrix}
  F_f(\alpha_f) \\
  F_b(\alpha_b) \\
  F_x(\kappa)
\end{bmatrix}
\]

\[
a_{fy} = \frac{F_f \cos(\delta_f)}{m_f}
\]
Assume on-board sensors for constructing state information

Only 3 accelerometers needed:
  -- x direction accel. of chassis
  -- lateral accel. at rear axle
  -- lateral accel. at front axle

Plus load cells at front & rear axles
  (to give estimate of c.g. location)

[Simulation used analytic proxies]
Criteria for various Design Scenarios:

1. Velocity Error
   • reduce to zero.
2. Velocity Rate
   • limit rate of velocity corrections.
3. Y-direction Error
   • reduce to zero.
4. Lateral front axle acceleration
   • re. “comfort” design specification.
5. Friction Sense [Proxy CONTEXT variable]
   • estimate closeness to knee of cof curves.
Utility Functions for three Design Scenarios:

[different combinations of above criteria]

1. U(1,2,3)
2. U(1,2,3,5)
3. U(1,2,3,4,5)

All applied to task of designing controller for autonomous 2-axle terrestrial vehicle.
Design Scenario 1.
Fewest restrictions on controller design.

\[ U_1 = -\frac{1}{2}(v_{err})^2 - \frac{1}{8}(v_{err})^2 - \frac{1}{16}(\dot{v})^2 \]

Velocity terms given lower coefficients, based on assumed specification that forward velocity be \textit{approximately} constant, with explicit specification for $Y$-error to become zero.
Design Scenario 1 Experimental Result

--> Step Velocity Change, via $U_1$:

Commanded change from 30 m/s to 40 m/s.
Design Scenario 1 Experimental Result

--> Lane Change $[\Delta y = 6m]$, via $U_1$

Tested with COF: Icy --> Car goes off the road!

[wasp trained on dry road, thus steering commands too aggressive.]
Design Scenario 2.

Add Criterion 5 ("friction sense") in U2. This is intended to

1.) allow aggressive lane changes on dry pavement, and

2.) make lane changes on icy road conditions as aggressively as the icy road will allow.

[Employ this as a Context Variable.]
[--- via the Utility function.]
Design Scenario 2.
Implement Criterion 5 via “Sliding Index”:

\[ SI = -10 \left( \frac{\partial a_y}{\partial \alpha_f} \right) \left( \frac{\partial a_y}{\partial \alpha_f} \right)_{\text{base}} \], where \( \left( \frac{\partial a_y}{\partial \alpha_f} \right)_{\text{base}} \) is the slope at very small slip angle (i.e., the linear portion of the curves).

SI takes on value of \(~10\) when sliding, and \(~0\) at no sliding.
Design Scenario 2

Resulting Utility Function $U_2$

$$U_2 = \begin{cases} U_1 & \text{for } SI < 3 \\ U_1 - .25SI^2 & \text{for } SI \geq 3 \end{cases}$$
Design Scenario 2 Experimental Result

--> Lane Change [$\Delta y = 6m$], via $U_2$:

Acceptable lane changes on both dry & icy pavement. Dry pavement response slightly slower than $U_1$, via SI penalty. Peak acceleration on dry pavement: $\sim .5g$

On ice, controller keeps request to lower limit allowed on ice.
**Generalize Test:** Ice patch at point where wheel angle is greatest in dry pavement case, retrained with coefficient modified.

![Graph showing Y Position and Wheel Angle over time](image-url)
Dry Lane Change to compare for Generalize test

Fig 6. Dry Lane Change

Y Position

Wheel

Time (s)
Design Scenario 3.
Add Criterion 4 ("comfort") in $U_3$.

[Limits Lateral front axle acceleration.]

$$U_3 = U_2 - \frac{1}{8}(a_f)^2$$
Design Scenario 3 Experimental Result

--> Lane Change \([\Delta y = 6m]\), via \(U_3\):

Lane change response for both road conditions slower, due to acceleration limiting term.
Conclusions from Utility Function Expts.

- Controller Designs resulting via DHP satisfy intuitive sense of being “good”.

- Control Engineer knows that controller design requires careful specification of objective, and that as change design criteria, controllers change.

- For DHP, control objectives are contained in the Utility Function.

- The DHP process embodied the different requirements for the three design scenarios in qualitatively distinct controllers -- all yielding intuitively “good” results, according to the design constraints.
Conclusions from Utility Function Expts., cont.

“Of particular interest to the present researchers is the ability of the DHP method to accept design criteria crafted into the Utility function by the human designers, and to then evolve a controller design whose response “looks and feels” like one a human designer might have designed.”
Example 3 (for DHP):
Aircraft Control Augmentation System
System configuration during DHP Design of Controller Augmentation System
Stick-x doublet: Pilot’s stick signal vs. augmented signal (the latter is sent to aircraft actuators).
• **Blue:** LoFLYTE® w/ **Un**augmented control
• **Red:** LoFLYTE® w/Augmented Control
• **Black:** LoFLYTE®*

Roll 1

(Double click window to play video.)
• **Blue**: LoFLYTE® w/ **Un**augmented control
• **Red**: LoFLYTE® w/ Augmented Control
• **Black**: LoFLYTE®*

Roll 2

(Double click window to play video.)
Pilot stick-x doublet signal (arbitrary scale in Fig), and roll-rate response of 3 aircraft: LoFLYTE® w/Unaugmented control, LoFLYTE® w/Augmented Control, and LoFLYTE®*. (Note: Responses of latter two essentially coincide.)
Roll-rate error (for above stick-x signal), between: LoFLYTE®* and LoFLYTE® w/Unaugmented Control, and LoFLYTE®* and LoFLYTE® w/Augmented Control signals
Higher Level application of ADP method.
Imagine:

1) Reaching down to do a gentle hand-shake with a little girl.
2) Putting out your hand to protect your fall just after stumbling going up a stairway.

Take mental note of differences in:

a) Speed
b) Force
c) Angles of elbow, wrist, palm, and fingers
d) Path of motions

All done “optimally” – in some sense.

How do we do it?
OUTLINE FOR REST OF PRESENTATION

Intelligent Control

Some foundational aspects to consider

- Context
  - High-level mini-history of control
- Context Discernment
- Experience-Based

- Early Experiments:
  - Direct via NN with Context Variable input
  - Via HLLA:
    - Identification
    - Control
Some basic components for Intelligent Control
OBSERVATION:
In the case of humans, the more knowledge / experience attained, the more improvement in effectiveness of performing new related tasks, with little or no speed penalty.

OBSERVATION:
In the case of AI rule-based systems, the more knowledge attained, the slower the processing.

CONCLUSION:
Need a different way to store and access experiential knowledge to approach human-level control capabilities.
Consider task of Driving a Car:
Example to provide basic idea hooks for rest of talk:
(Assume *experienced* car driver)

I. Car attributes:
   1) driving own car;  2) driving friend’s car.

II. Environment: clear afternoon with
   1) dry pavement;  2) icy pavement.

III. Performance criteria (wrt Task/Objectives):
   1) Road race: minimize time.
   2) Elderly relative on excursion: maximize comfort.

→ Driver uses same base set of driving skills, but when change from #1 to #2, make adjustments to “control law” and/or “decision logic, from a collection previously acquired via *EXPERIENCE*.

*CONTEXT* comprises I, II, & III.]
CONTEXT:
We formulate context as comprising three components: A) Plant, B) Environment, and C) Objectives (via associated performance criteria – labeled CF). Specification of all three yields a specific context; to each specific context there corresponds a particular control law; a change in any of the components results in a different context.
Since **CONTEXT** is fundamental to the approach, a historical overview of the control field was performed vis-à-vis the explicit role that **context** has (or has not) played in the various formulations and approaches.
Overview of Controls Field vis-à-vis CONTEXT -

Described as Four Phases:

**Phase 1**: DESIGN BASED on INTUITION and 

**Phase 2**: DESIGN BASED on MATHEMATICAL TOOLS. 

e.g. Fourier and Laplace, state space, stochastic, Hilbert space, algebraic and geometric topological methods…

e.g. Classical Control, Modern Control, Optimal Control, Stochastic Control, and Robust Control
Overview of Controls Field, cont.

- **Phase 3**: DESIGN for CONTEXT DEPENDENCE.
  
e.g. Adaptive Control and Learning Control

- **Phase 4**: [NEW] DESIGN for EXPERIENCE-BASED PROCESSES
  
  including autonomous Context Discernment and Model Selection
Stipulated Requirements for the [new] Phase 4:

Agent has the ability to:

a) use *experience* for model selection (plant or controller); and

b) do so *effectively* and *efficiently*. 
Fundamental aspects to consider:

a) *context*, b) *discerning* current context, 
c) *selecting* appropriate model from *experience repository* for the discerned context, and 
d) doing the latter two in an *effective* and *efficient* manner.
General Example of Experience Based Ideas:
Quadruped Robot (AIBO Dog)
To move beyond working with just simulations, we performed initial stages of implementing a Robot with Context Discerning and Controller Selecting capability at NWCIL.

Sony AIBO robotic dog has been modified into a research platform.

It’s task is to learn to discern changes in walking surface types (e.g.: carpet, hardwood, inclines, etc.), and adjust its gait accordingly.
AIBO experiments to date:

1. Constructed five different surface types (4’ long)
   [hardwood, thin foam, thin carpet, thick shag carpet
    reversed shag carpet]

2. Used genetic algorithm to develop “good” AIBO gait for each of five surfaces.
   [Note: each resulting gait yields better performance measure on its respective surface than does the AIBO default gait; differences are visually discernable.]

3. AIBO made test walks on the five surfaces for each gait, and data streams from 17 joint-actuator sensors were noted [“Kinesthetic Experience”].
AIBO experiments (cont.):

5. When walking on a surface, AIBO discerns the surface type (CONTEXT) by processing its current “kinesthestic experience” through the models in its repository, and selects the most similar one.

6. It then selects the gait corresponding to this surface, and adjusts its walk accordingly.

7. Show video clips...
AIBO experiences change of surface type.

Decisions based only on ‘kinesthetic experience’ vector. (Double click window to play video.)
AIBO experiences change of surface type.

Decisions based only on ‘kinesthetic experience’ vector. (Double click window to play video.)
AIBO experiences change of surface incline.

Decisions based only on ‘kinesthetic experience’ vector.  
(Double click window to play video.)
AIBO experiences change of surface incline.

Decisions based only on ‘kinesthetic experience’ vector. (Double click window to play video.)
Phase 4 controller design via ADP

**Approach 1: Direct access to experience in NN repository.**

i) Embed multiple memories (“experience”) in NN, and index them via Context Variable(s) input.

ii) This entails predefining mechanism(s) for context *discernment* and employ as “Context Variable” (CV) input to NN controller during ADP training.

iii) During operation, the NN has learned to employ the CV input as “index” to the embedded distinct controller designs for different contexts.
Standard Use of ADP

Single Controller in NN Designed/Trained via ADP

For Phase 4 Approach 1:
NN as Repository, with Context Variable (CV) as Selector

NN Designed via ADP with CV results in multiple embedded controllers.
[In operation, CV serves as SELECTOR for the different Controllers.]
**Example 3a** (Airplane controller to Include experience/context via DHP):

1) NN as Repository, with Context Variable (CV) as Selector

Aircraft Controller enhanced to also accommodate C.G. shift (via DHP).

[Predefined Context Variable is c.g. location.]
[Calculated while in flight -- realistic].
Experiment conditions for c.g. shift as change of context:

- LoFlyte® hypersonic aircraft shape. Nominal c.g. at 50% of its length.
- At ~ 56% of length (ftb), aircraft becomes open-loop unstable.
- For c.g. location > 56%, critical task of controller is to keep plane (closed-loop) stable.
- Devised method to (realistically) calculate current c.g. location (i.e., method of context discernment).
- Used this as Context Variable input to NN during ADP training.
- NN learned to employ CV as “index” to controllers that keep plane stable as changes in c.g. are instantiated.
- Trained to 57% c.g. shift. During tests, NN did great for c.g. locations between 56% - 57% (unstable regions), plus, extrapolated out to 58%.
- Show simulated video clip. NN controller closely tracks ideal aircraft whose c.g. is assumed not to have changed, and standard controller yields severe nose-up response.
• **Blue**: LoFLYTE® w/ *Un*augmented control
• **Red**: LoFLYTE® w/Augmented Control
• **Black**: LoFLYTE®*

(Double click window to play video.)
Conclusions re. Aircraft Control Augmentation Example:

- **Customer’s motivation** for this work was a desire to generate a non-linear controller that has control capabilities equivalent to that of an “experienced pilot” with respect to specified sudden changes in aircraft “characteristics”. C.g. shift was the characteristic shift stipulated by the customer for the project.

- **Research objective** included desire to demonstrate application of “experience based” notions – specifically to demonstrate NN as Repository with Context Variable (CV) as Selector.

- **Successful base demonstration of both goals was achieved.**
Recall Stipulations for Phase 4 Controls:

Agent has the ability to:
   a) use experience as basis for selection; and b) do so effectively and efficiently.

Fundamental aspects to consider:
   a) context, b) discerning current context,
   c) selecting appropriate model from experience repository for the discerned context,
   d) doing the latter two in an effective and efficient manner.

**Aircraft Experiment: Success with all above components.**

Given aircraft specified as Context for the control task.

Change of aircraft characteristics = change in context.

Discernment performed via predefined CV.

“Experience” embedded in NN via training.

NN selected appropriate model from repository for discerned context, and
NN correctly and quickly made switch (selection) to new controller when c.g. shifted.
Phase 4 controller design via ADP

Approach 2:
Employ ADP methods to
develop means for performing selection of controller from a repository instantiated differently than in Approach 1

Employ ADP methods to
implement Context Discernment process (includes system identification for control application).

An ADP method called HLLA is employed for both.
KEY IDEA of HLLA:

- Re-purpose the Reinforcement Learning method (to a “higher level”) such that
- Instead of using it to *design* an optimal controller for a given task,
- An already achieved *collection* of such solutions for a variety of related contexts is provided (as an *experience repository*), and
- HLLA creates a strategy for optimally *selecting* a solution from the repository.
Phase 4 controller design via ADP–Approach 2 (HLLA)

[HLLA creates a strategy for optimally selecting a solution from the repository.]

Research Stage 1: System Identification Task

Here, the Repository comprises collection of plant models (in contrast to controller models), and task is to optimally select a model that matches plant being identified.
HLLA Stage 1, **Example 1: Pole-Cart Problem** (Revisited)

**Assume:**

1) A controller for nominal Pole-Cart is in operation.
2) Pole-Cart attributes change suddenly.
3) For controller to “adapt”, needs to find present condition of the Pole-Cart. [Context Discernment; System Identification.]

**Method:**

a) Craft a “repository” of various versions of the Pole-Cart plant.
b) Develop HLLA process to optimally select (wrt quality of controller, and speed of access) model from the repository that matches current plant condition.
HLLA System Identification (SysID) for Pole-Cart, cont.

Approach taken:

a) Employed equations of motion of Pole-Cart plant as the “repository”. Changes in plant accomplished via changes in parameter values of the equations.

b) Form of equations employed had mass and length of pole among its parameters. This enabled creation of “plant manifold” [see appendix] and defining a coordinate space in terms of the parameters.

[Description of overall HLLA process ➔ next slide:]
HLLA Overview:

“Controller” role for ADP

“Plant” role for ADP

\[ \Delta C_D(t) \]

\[ C_D(t) \]

\[ R_A(t) \]

\[ z^{-1} \]

CDN (Context Discerning Network)

Model (With Context Variables \( C_D \))

System (With Context Variables \( C_A \))

\[ D(t) = R_A(t+1) - R_D(t+1) \]

\[ U(t) = (R_A(t+1) - R_D(t+1))^2 \]

\[ \lambda(t) \]

Used To Train CDN And Critic

Used To Train Critic
HLLA SysID for **Pole-Cart** – Experimental Results:

**TOP:** Context Discernment in response to change in plant parameter values (context change) at every 50th iteration.

**BOTTOM:** Errors between values for pole-cart system state variable and for models selected during discernment process.
HLLA Stage 1, Example 2: Neural Network as Plant

Approach taken:

a) Crafted a neural network of specified structure and element type (MLP) to be "repository". Changes in plant accomplished via changes in selected weight values of NN.

b) Form of equations associated with MLP have weights as "parameters". This enabled creation of "plant manifold" [see appendix] and defining a coordinate space in terms of the weights/parameters.

[Description of overall HLLA process  → next slide:]
**Neural Network as Plant**: Experiment details

- NN parameters (weights and biases) are continuous.
- Connection weights are held fixed, bias weights allowed to vary.
- Each setting of the bias weights defines a unique NN.
- Generate a set of such NNs by varying the bias weight values.
- Let the biases be the coordinates $\rightarrow$ neural manifold (see Appendix).
- Procedure is equivalent to one used for Pole Cart (cf. HLLA overview slide).
- Context Discerner’s task: Determine NN in repository that corresponds to the specific one currently instantiated in the System box.
- Included NNs with different ratios of fixed to variable parameters, with good success involving some 40 parameters, and ratios of about five to one.
- Observe that at each change of NN in the System box, the context discerner successfully noted change and rapidly selected the corresponding NN from the repository.
- Error plot demonstrates a spike in error value when the change occurs, followed by a quick recovery – 10 or fewer observations of context used.
HLLA SysID for NN as Plant – Experimental Results:

**TOP:** Context Discernment in response to change in NN plant parameter values (context change) at every 100th iteration.

**BOTTOM:** Errors between values for pole-cart system state variable and for models selected during discernment process.
Proposed model of Experience-Based Control Configuration

Starting Condition:

- CONTROLLER

Perform Controller SELECTION (EB)

EB-UPDATED CONTROLLER MODEL

Run Simulation

EB-UPDATED PLANT MODEL

Criterion Function Assessor (CFA)

Context Monitoring

All OK

Off Nominal

Off Nominal

All OK

Install Updated Controller Design

UPDATED PLANT MODEL

Perform (EB) SID

UPDATED PLANT MODEL

Proposed model of Experience-Based Control Configuration
OVERALL CONCLUSION ABOUT EB APPROACH

I conjecture that the proposed experience-based approach will usher in a whole new phase of development of the decision and controls fields – making a significant stride toward the achievement of more human-like decision and control.

Also conjecture that the context discernment concepts plus the manifolds representation will provide a basis for constructing learning agents capable of long term rapidly accessible memory. If so, this could pave the way for scaling neural systems to brain-like capabilities.
QUESTIONS?
APPENDIX 1: Definitions
Agent: computational intelligence device (that, in this paper, is to perform the acts of context discernment and selection, along with possible design refinement).

Context Variables (Agent centric): those attributes of i) the environment and ii) the plant/process whose variations could engender changes to the decision rule / control policy employed by the Agent while accomplishing the Agent’s current objective or goal; and in addition, iii) the criteria (representing the objective or goal) to be used for designing and subsequent selection of the decision rule or control law. [The term Criterion Function (CF) is used here to represent these criteria.]

Context Space (Agent centric): a vector space in which each context variable is assigned to a dimension. The Context Space concept comprises three sub-spaces; one each associated with the i) Plant, ii) Environment, and iii) Criterion Function.

Context (Agent centric): a point in Context Space; the set of values taken on by the context variables in a given situation.

Context Awareness: the act of monitoring the application to take notice (become aware) that a change may be occurring in the Context.

Context Discernment: the act or process of determining the current values of the context variables (current point in Context Space) appropriate to the task being performed. [Webster on-line for ‘discern’: to recognize or identify as separate and distinct.]

Experience-Based approach: A two-component concept:

Component A: Repository of previously developed context-specific models (controller or plant models), and

Component B: Algorithms used by the Agent to effectively and efficiently select a model from the repository as changes in context occur. [Note: A key task of the Higher Level Learning Algorithm (defined below) is to train the Agent to learn Component B.]
Selection: the act of choosing/retrieving an appropriate element of the repository corresponding to the discerned context.

Higher-Level Learning Algorithm (HLLA): The reference level for the term ‘higher’ is the case where the learning algorithms are applied directly to the design of optimal controllers (as in Learning Control), ones that would be accumulated in the repository (c.f. Fig. 1). ‘Higher-Level’ here means applying the learning method to create a strategy for selecting an appropriate controller from the repository, where the process of selection is optimized; thus, the ‘focus’ of the learning process is at the “next level up”. Definition of the Utility function (a specific type of CF) is key for application of this process. Note: When the Contextual Hierarchy ideas mentioned in Section I are developed, more levels will be involved.

World Space (Agent Centric): A vector space whose dimensions are associated to designated attributes of the Agent’s relevant environment, its physical body, and the external CF.

[Note: This definition is included for completeness. It is not explicitly used in this paper, but is used in related publications in terms of mappings from World Space to Context Space, e.g. [39].]

Guidelines:
Parametric models/equations are used to represent the Plant, Criterion Function (CF), and Environment (for the latter, measurements may serve as parameters w/o an explicit model). Construct (conceptually) a Parameter Space that comprises three sub-spaces: (Plant, Environment, CF). The associated parameters serve as Context variables for the discernment activity; Agent’s Context Space may be a sub-space of Parameter Space. Controllers are also represented via parametric models.
APPENDIX 2: Manifolds
Geometric Topology *manifolds* construct provides a useful formalism:

A manifold is:

1) a set of elements, $S$, and
2) a coordinate system

(a one-to-one mapping from $S$ to $\mathbb{R}^n$ that specifies each element in $S$ via a vector of $n$ real numbers, a.k.a. the *coordinates* of the element).

We let the *experience repository* be the set portion of a manifold.

The manifold’s *coordinate space* serves as a *searchable indexing vehicle* for the repository, and since the coordinate space is $\mathbb{R}^n$, the Euclidean distance provides a natural metric for “nearness”.
Example manifold construction:

- Let the manifold set be a collection of neural networks (NNs) generated via a NN whose structure is fixed,
- and its adjustable parameters (weights and biases) are made to take on all possible value combinations. Each such combination yields a distinct member of the set,
- and the parameter values may serve as the coordinates;
- a point in the coordinate space may be called the set element’s “address”.

• When the manifold’s set are NNs, we use the label *neural manifold*
  
  – In this case, care is needed relative to two aspects:
    1) while each coordinate point corresponds to a distinct NN instantiation, nevertheless, many such points may all perform the same mapping, and
    2) the set of (distinct) mappings that can be performed by this set of NNs is typically a subset of all possible mappings on the NN’s input domain to its output range (called the NN’s Performance Subset).

• Mapping from Context Space to the policy manifold may in general be many-to-one
  
  (in the controls vocabulary, changes in the plant dynamics or in its environment do not necc. imply a needed change in control policy).
The indexing schemas for both a plant and policy neural manifold may employ the weights of their respective set of NNs. So far so good.

But, how does one go about crafting a mapping between, say, the plant manifold’s coordinate space to that of the policy manifold?

Clearly, such a mapping will be required for the Agent to select a policy based on information about the plant model.

More generally, how does one craft an appropriate mapping from the full Context Space (whatever form of representation is employed) to the coordinate system of the policy manifold?

The task of answering these questions is assigned to a Higher Level Learning Algorithm (HLLA) – i.e., the answers are to be learned.
For another aspect of mappings, consider a **linear plant example**, and assume the plant transfer functions in the plant manifold are factored polynomials but the CF requirements are given in terms of an expanded polynomial representation (e.g., if requirements are given in terms of damping coefficient for a second order system).

While the two representations are equivalent, the Agent would need a mapping between the two to accomplish the controller selection (e.g., via factoring the polynomial, or equivalently, multiplying out the factored polynomial).

In the second order case, the notion of nearness in the CF sub-space would be in terms of the damping coefficient, whereas in the corresponding plant manifold coordinate space, nearness would be in terms of S-plane pole locations.
• As an intuitive example of notions such as *efficiency*, *nearness*, and *mappings*, consider the example of a store that rents movie DVDs …

• … shelved alphabetically or by content type

• … Which is more *efficient* for the customer depends on the customer’s needs and knowledge.
APPENDIX 3: Future Explorations
As noted in (Lendaris, 2008), there are many issues to still confront as the idealizations in the experience-based notions are relaxed, little by little. In the examples provided in Sections 4 & 5, the repositories were populated with models of the precise form of the plant being identified, and similarly for work related to controller selection. What is to happen as the plant and controller models are close approximations but not exact? What is to happen when noise is allowed into the process? The notion of “levels” is invoked in the experience-based notions; while these concepts are more typically used in the systems literature, cf. (Lendaris, 1986), it will be useful to more deeply explore their application to the suggested Experience-Based Control notions.
Key exploration steps ahead:

• Refine and further develop ideas related to Context Discernment / System Identification thus far developed.

• Move into the Controller Selection aspect of the suggested EB Control method.

• Expand the exploration to multiple-level considerations.

• Provide one or more feasibility demonstrations in support of developing theory and techniques for populating the experience repositories – progressing from the synthetic methods already demonstrated to more and more general ones.
Key exploration steps ahead (cont.):

• Formalize ideas about and develop demonstration experiments for incorporating application domain knowledge as *repository constraints* – of a nature that facilitates the larger objectives of improved speed of access and good generalization.
Key exploration steps ahead (cont.):

- Formalize ideas about constructing and achieving needed mappings between components of Context and from Context to Repository, and develop demonstration examples useful for theory development.
Key exploration steps ahead (cont.):

• Further develop ideas related to multi-level aspects of EB Identification & Control, leading to a Context Space Hierarchy notion, and use the associated ideas as a guide in refining and further defining the HLLA concepts and training methods.

• Formalize and further develop the role of the human designer in providing “higher level” knowledge for crafting the RL process entailed in the HLLAs, particularly, designation of state variables and CF’s specialized to the multi-level conceptualization.
Key exploration steps ahead (cont.):

• Develop demonstration examples for EB controls, similar to the successful demonstrations of the Context Discernment (systems identification) part of the EB process thus far accomplished.