4-8-2011

Higher-level Application of Adaptive Dynamic Programming/reinforcement Learning – A Next phase for Controls and System Identification?

George G. Lendaris
Portland State University, lendaris@sysc.pdx.edu

Higher-Level Application of Adaptive Dynamic Programming/Reinforcement Learning – a Next Phase for Controls and System Identification?

Keynote Talk at
2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning
Paris, April 12, 2011

George G. Lendaris ©
NW Computational Intelligence Laboratory
Systems Science Graduate Program
Portland State University, Portland, OR

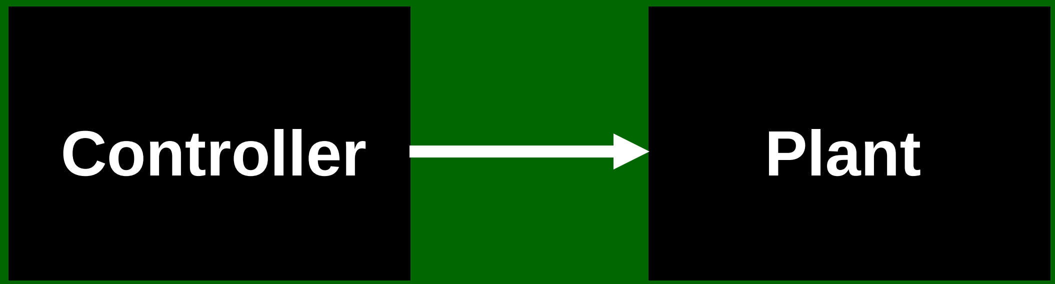
Order of presentation in this talk:

1. Controls
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. **Higher-Level Application of ADP** (to controls)
6. to System Identification
7. Examples
8. Concluding comments

Order of presentation in this talk:

1. Controls [→ Human-like Controls]
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. **Higher-Level Application of ADP** (to controls)
6. to System Identification
7. Examples
8. Concluding comments

Basic Control Scenario:



Problem statement: For a given plant in a given environment, design a controller to achieve stated design objectives / success criteria.

In context of this Symposium:
Design of the controller is via Adaptive Dynamic Programming / Reinforcement Learning methods

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):

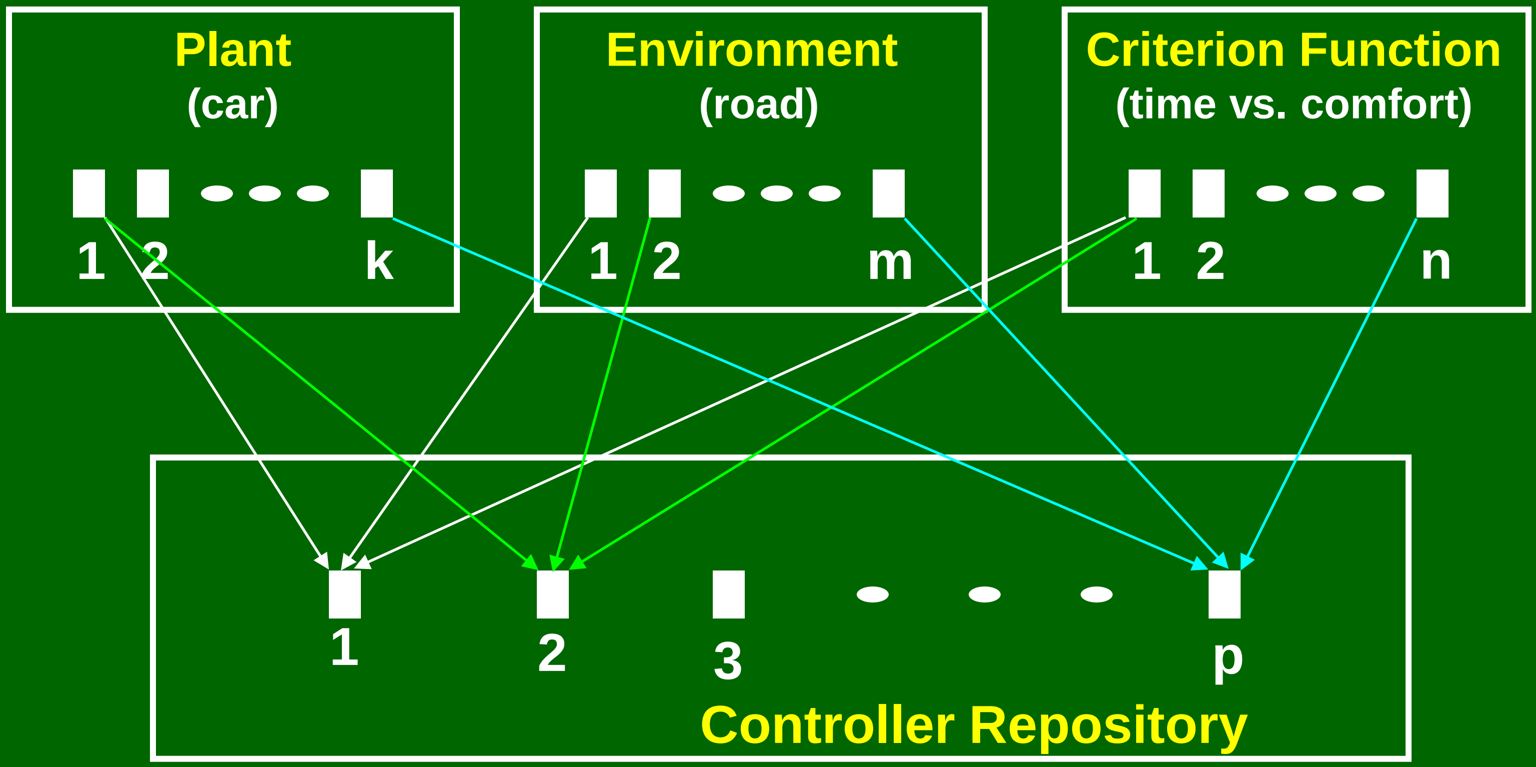
1) Road race: minimize time.

2) Elderly relative on excursion: maximize comfort.

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

[**CONTEXT** comprises I, II, & III.]

Basic Control Scenario, cont.:

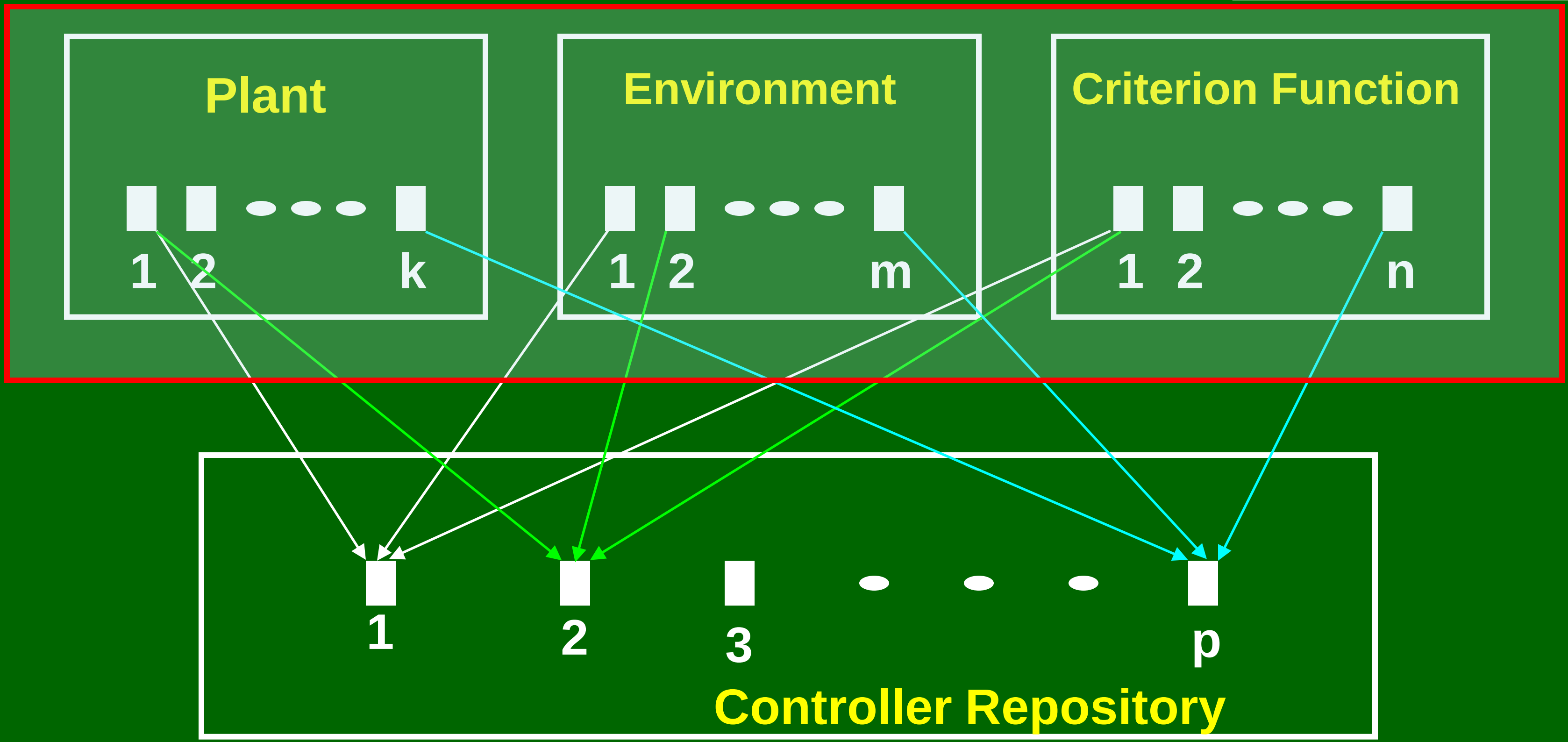


Designer of controller needs following:

- Problem domain specifications, including all available *a priori* and current information about Plant and Environment
- Design objectives / Criteria for “success” → (Criterion Function)

Context

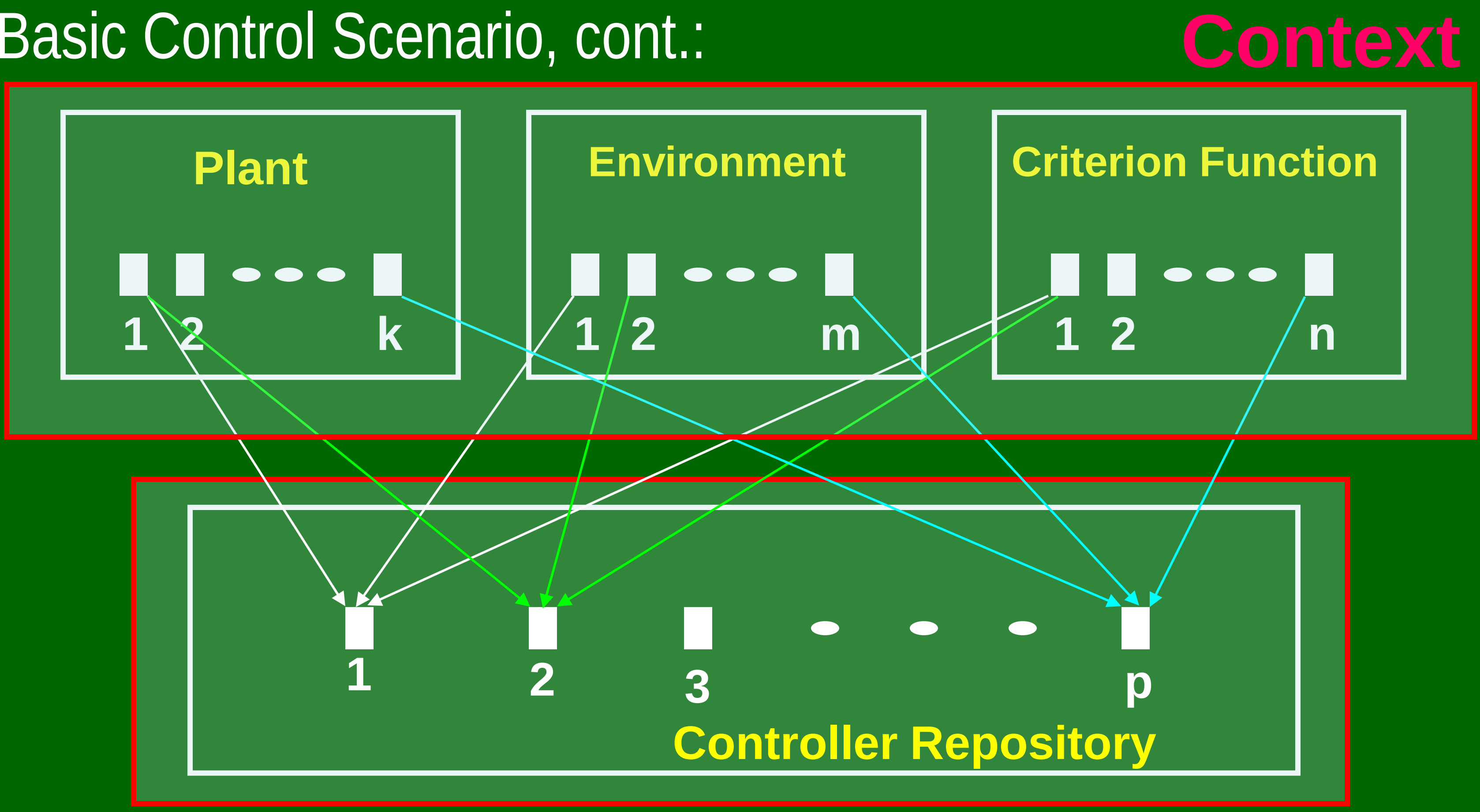
Basic Control Scenario, cont.:



Designer of controller needs following:

- Problem domain specifications, including all available *a priori* and current information about Plant and Environment
- Design objectives / Criteria for "success" → (Criterion Function)

Basic Control Scenario, cont.:



Experience

Designer of controller needs following:

- Problem domain specifications, including all available *a priori* and current information about Plant and Environment
- Design objectives / Criteria for "success" → (Criterion Function)

Human-Like Control

Imagine two different scenarios:

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 of hand movement
b) FORCE of hand contact
c) ANGLES of elbow, wrist, palm, and fingers
d) Path of motions

All selected “optimally” – in some sense.

HOW DO WE DO IT?
HOW ROOTED IN EXPERIENCE?

OBSERVATION 1:

In the case of humans, the more knowledge / *experience* attained, the more improvement in effectiveness of performing new related tasks, and with enhanced speed of execution.

OBSERVATION 2:

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.

Order of presentation in this talk:

1. Controls [→ Human-like Controls]
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. **Higher-Level Application of ADP** (to controls)
6. to System Identification
7. Examples
8. Concluding comments

- **Reinforcement Learning:**

  A type of learning by an agent where the environment provides qualitative feedback about its actions, and the agent's next actions strive to maximize some type of long-term "reward" ["reinforcement", utility function].

- **Adaptive Critic** type of Reinforcement Learning:

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

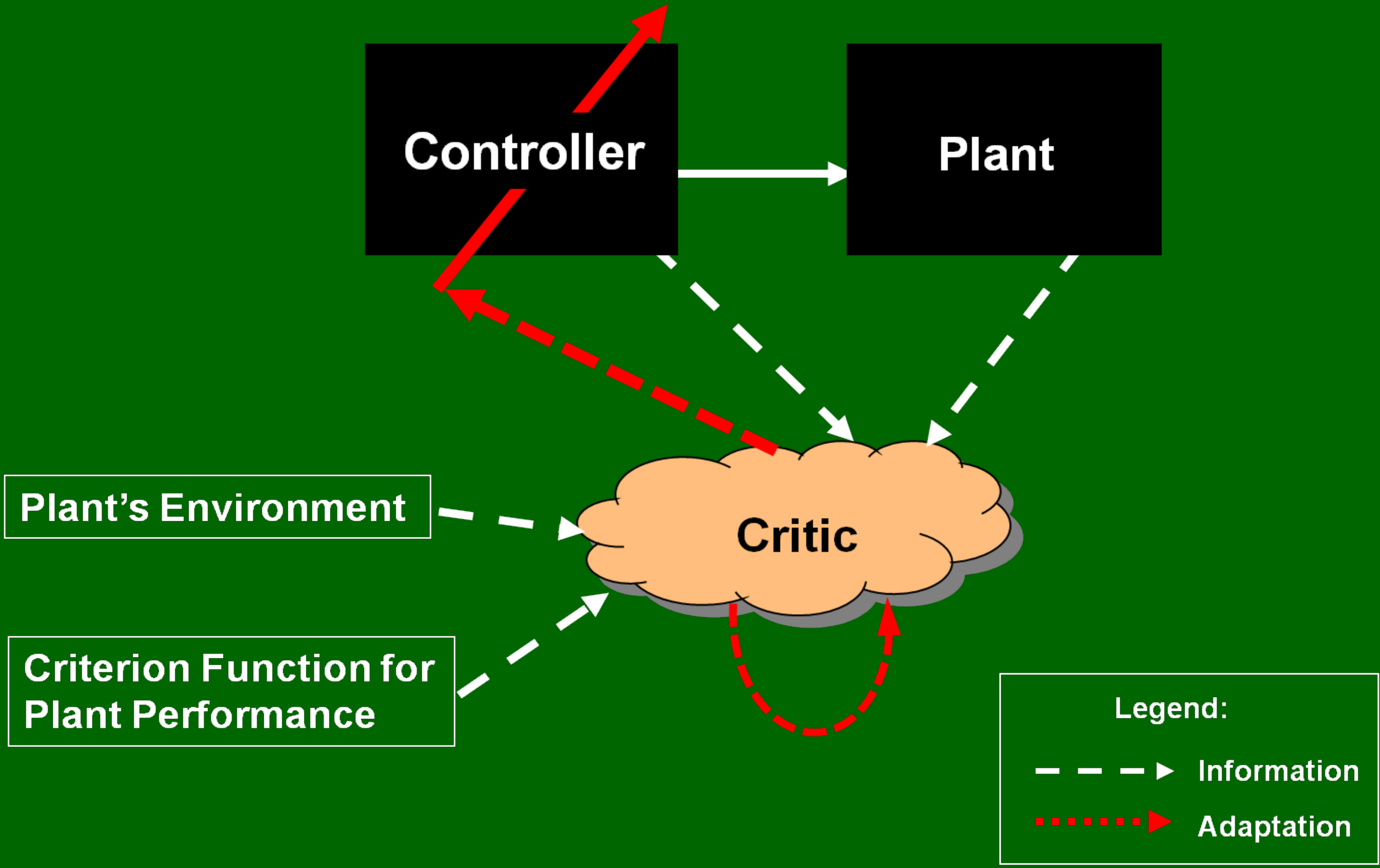
- **Implementation** of Adaptive Critic method:

  May be implemented using two learning agents (e.g., *neural networks* , *Fuzzy systems*):

  ---> *one in role of controller*, and

  ---> one in role of *critic*.

Overview of Adaptive Critic approach:



Order of presentation in this talk:

1. Controls (including some historical aspects)
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. **Higher-Level Application of ADP** (to controls)
6. to System Identification
7. Examples
8. Concluding comments

Dynamic Programming:

- Principled method for determining optimal control policies for discrete-time dynamic systems.
- Transition via control *u* [from state *R*(t) to *R*(t+1)] at a cost *U*.
- Optimality is defined in terms of minimizing the sum of all the costs ('cost-to-go') to be incurred while progressing from any state to the end state.
- Objective of DP is to calculate numerically the optimal cost-to-go function *J** and its associated optimal control policy.

Dynamic Programming:

- User provides the Design Objectives / Criteria for "success"

  through a Utility Function, U(R(t), u(t)) [local cost]

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

$$J(R(t), u(t)) = \sum_{k=0 \rightarrow \infty} \gamma^k U(t+k)$$

[cost-to-go]
[value function]

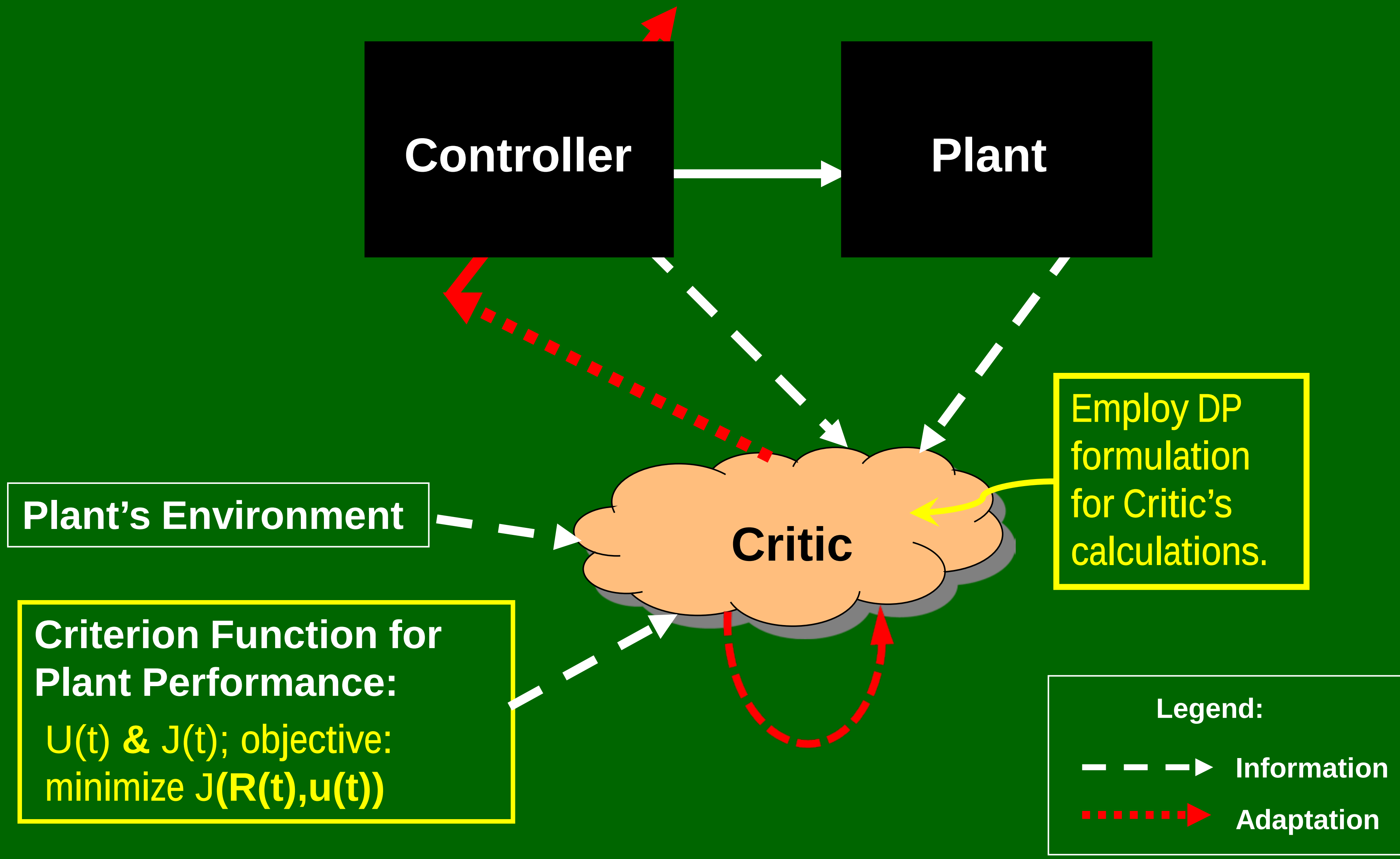
- Objective is to minimize J(R(t), u(t))

Important side note: $J(t) = U(t) + \gamma J(t+1)$ [Bellman Recursion]

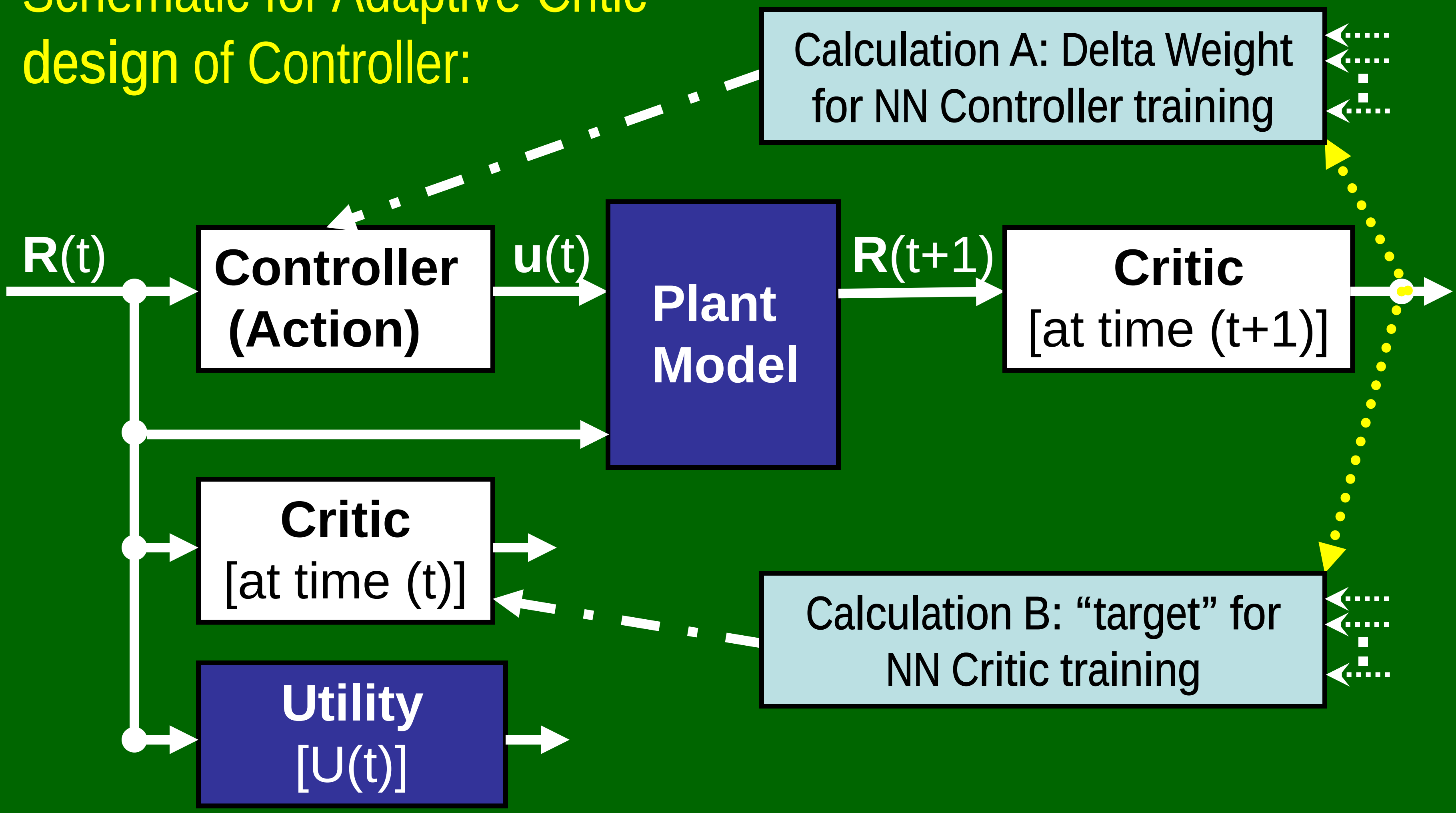
Order of presentation in this talk:

1. Controls (including some historical aspects)
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. **Higher-Level Application of ADP** (to controls)
6. to System Identification
7. Examples
8. Concluding comments

Revisit Overview of Adaptive Critic approach:



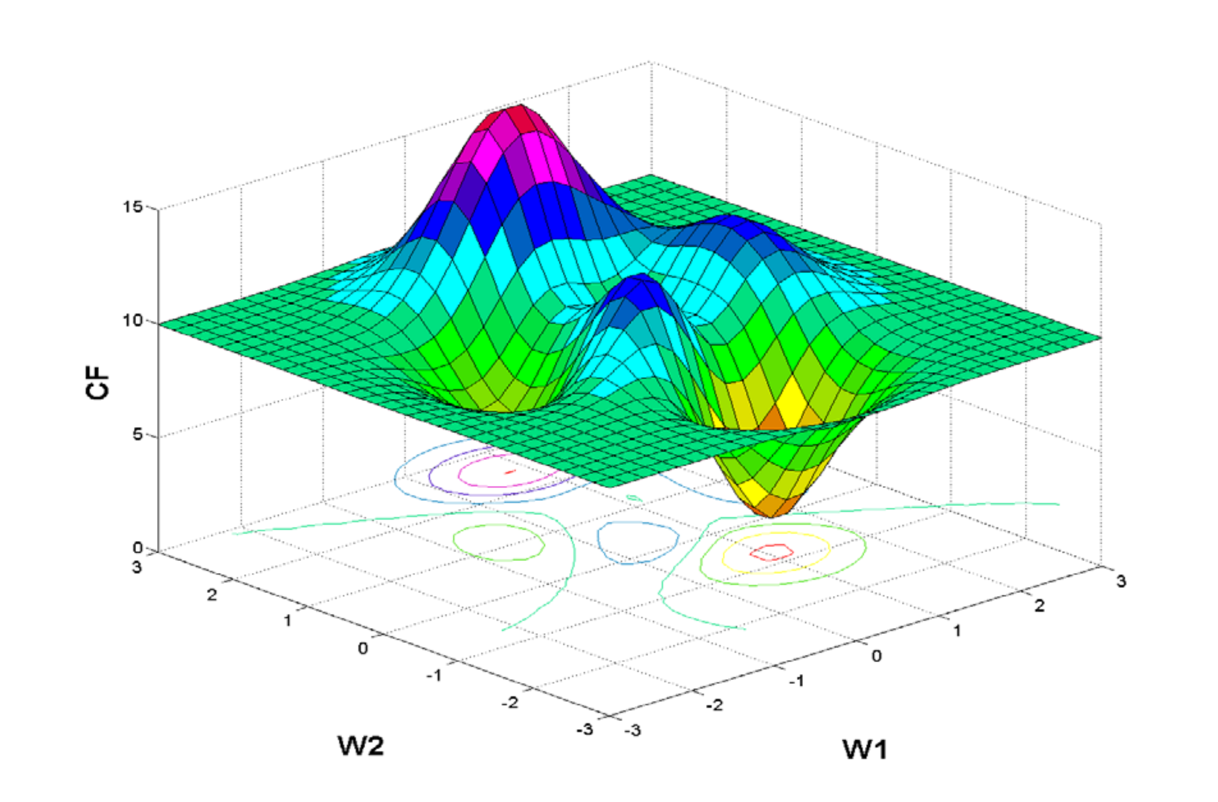
Schematic for Adaptive Critic design of Controller:



Dark Blue Boxes: analytic expressions. Medium Blue Boxes: critical calculations. White Boxes: learning agents (e.g., NN, Fuzzy, etc.).

Mathematical approach:

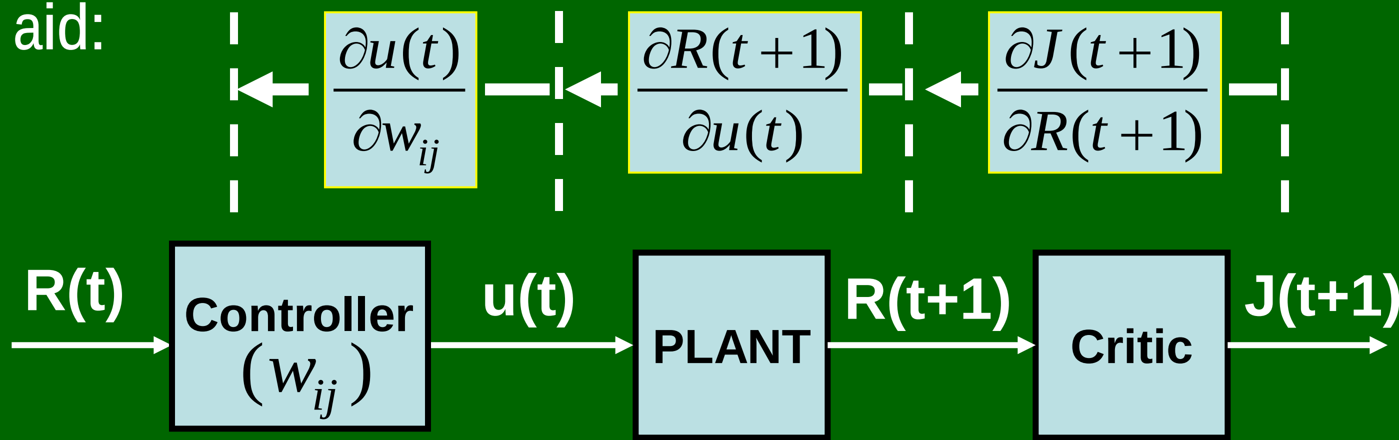
Perform gradient descent on a surface representing Bellman's J function constructed in NN controller's weight space.



Employ Gradient Descent approach to develop “Delta Rule” for controller’s weights w_{ij} to minimize cost-to-go J.

Characterize Gradient Descent via $\frac{\partial J(t)}{\partial w_{ij}(t)}$ and employ the chain rule of differentiation to evaluate it.

Visualization aid:



Available to us:

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

Define Delta Rule for weights in controller NN (via Gradient Descent):

$$\Delta w_{ij}(t) = -lcoef \cdot \frac{\partial J(t)}{\partial w_{ij}(t)} \qquad (1)$$

Invoke chain rule

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

Invoke Bellman Recursion: J(t) = U(t) + γJ(t + 1)

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

and

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

Let $\lambda_s(t+1)$ *represent this term.*

Summarizing, it follows that Controller training is based on:

$$\frac{\partial J(t)}{\partial u_k(t)} = \frac{\partial U(t)}{\partial u_k(t)} + \sum_{s=1}^{n} \frac{\partial J(t+1)}{\partial R_s(t+1)} \cdot \frac{\partial R_s(t+1)}{\partial u_k(t)} \qquad (5)$$

Via CRITIC

Via Plant Model

Similarly, Critic training is based on:

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

Via CRITIC

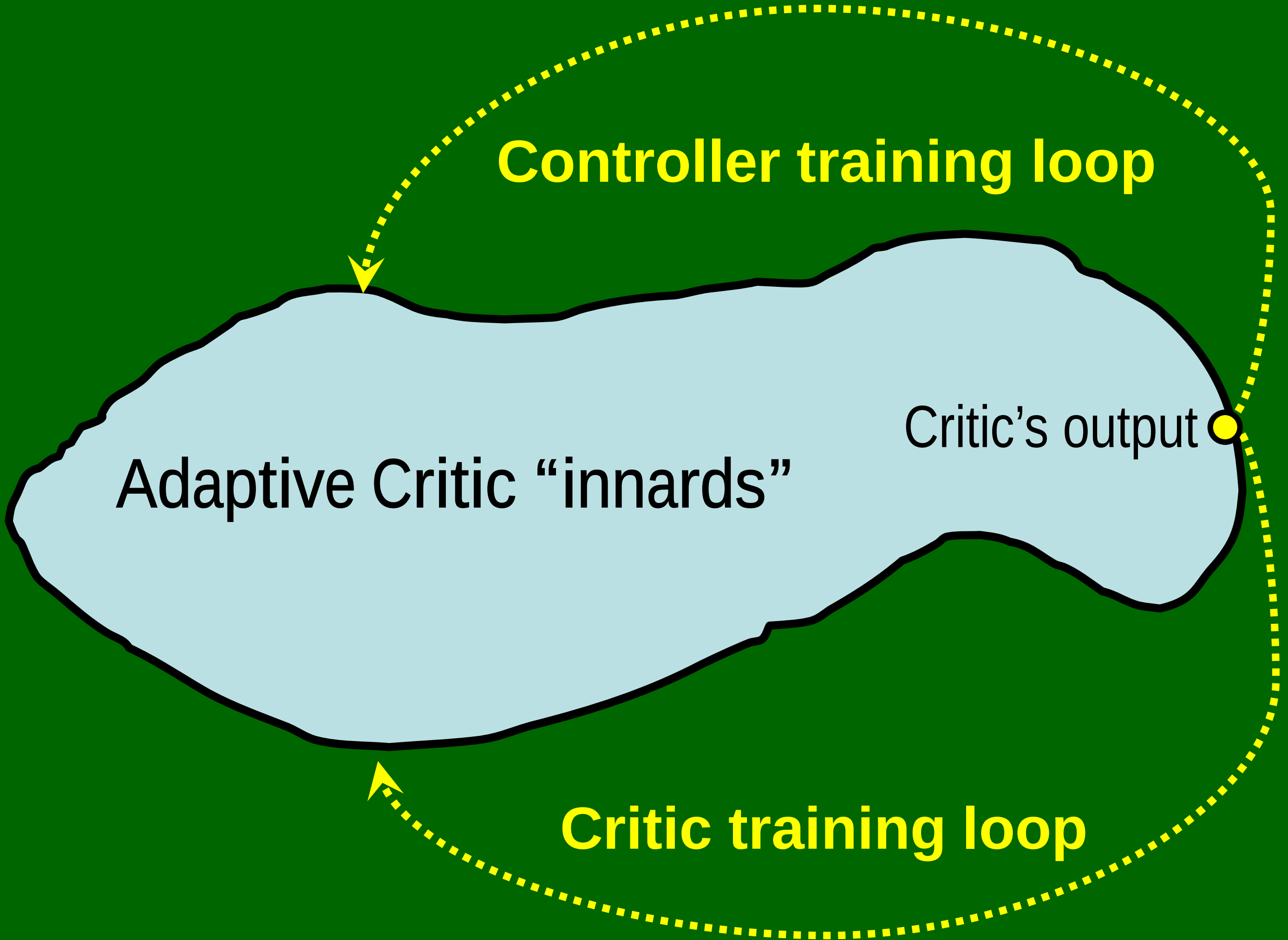
Via Plant Model

Via Controller

[Bellman Recursion & Chain Rule used in above.]

Plant model is needed to calculate partial derivatives for DHP …

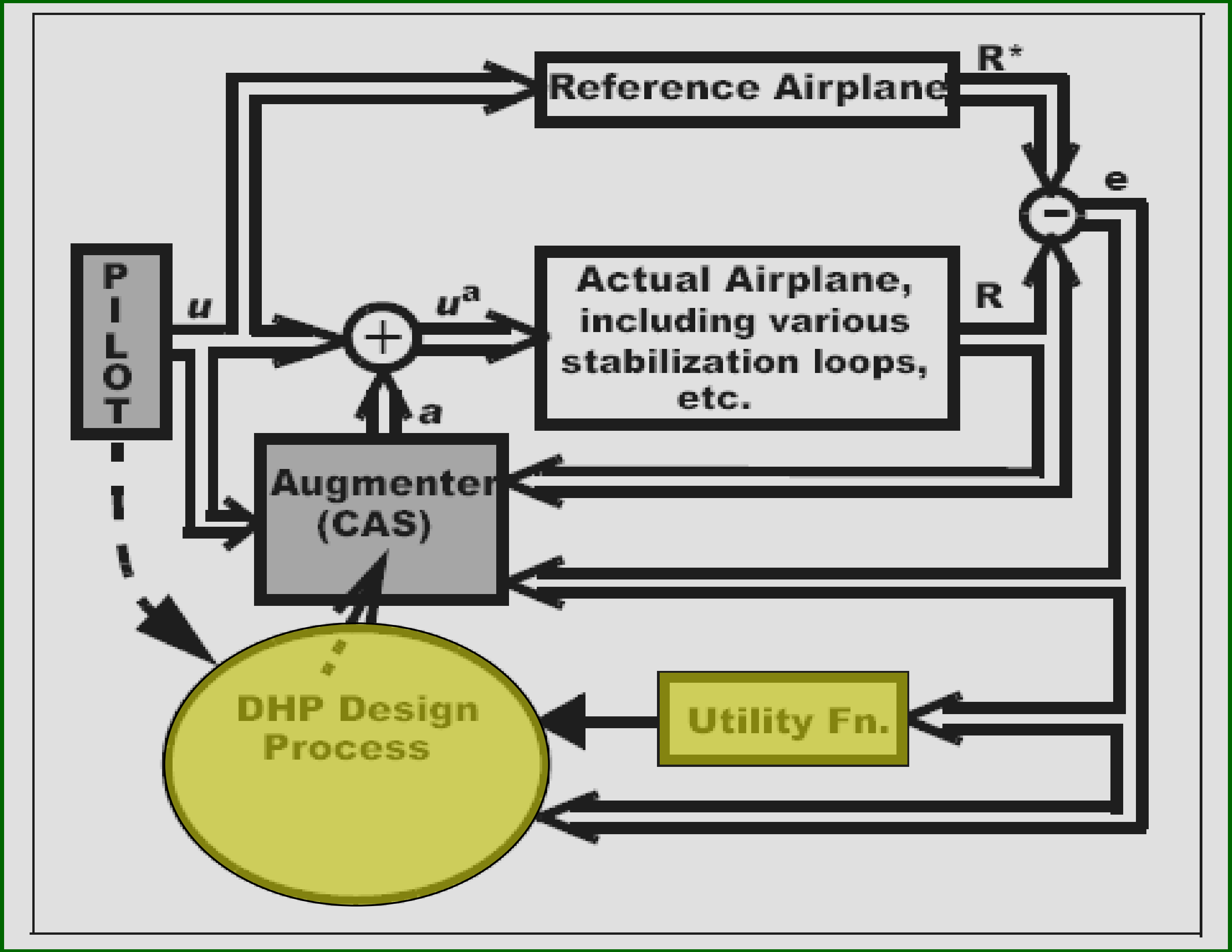
Two train loops in Adaptive Critic method:

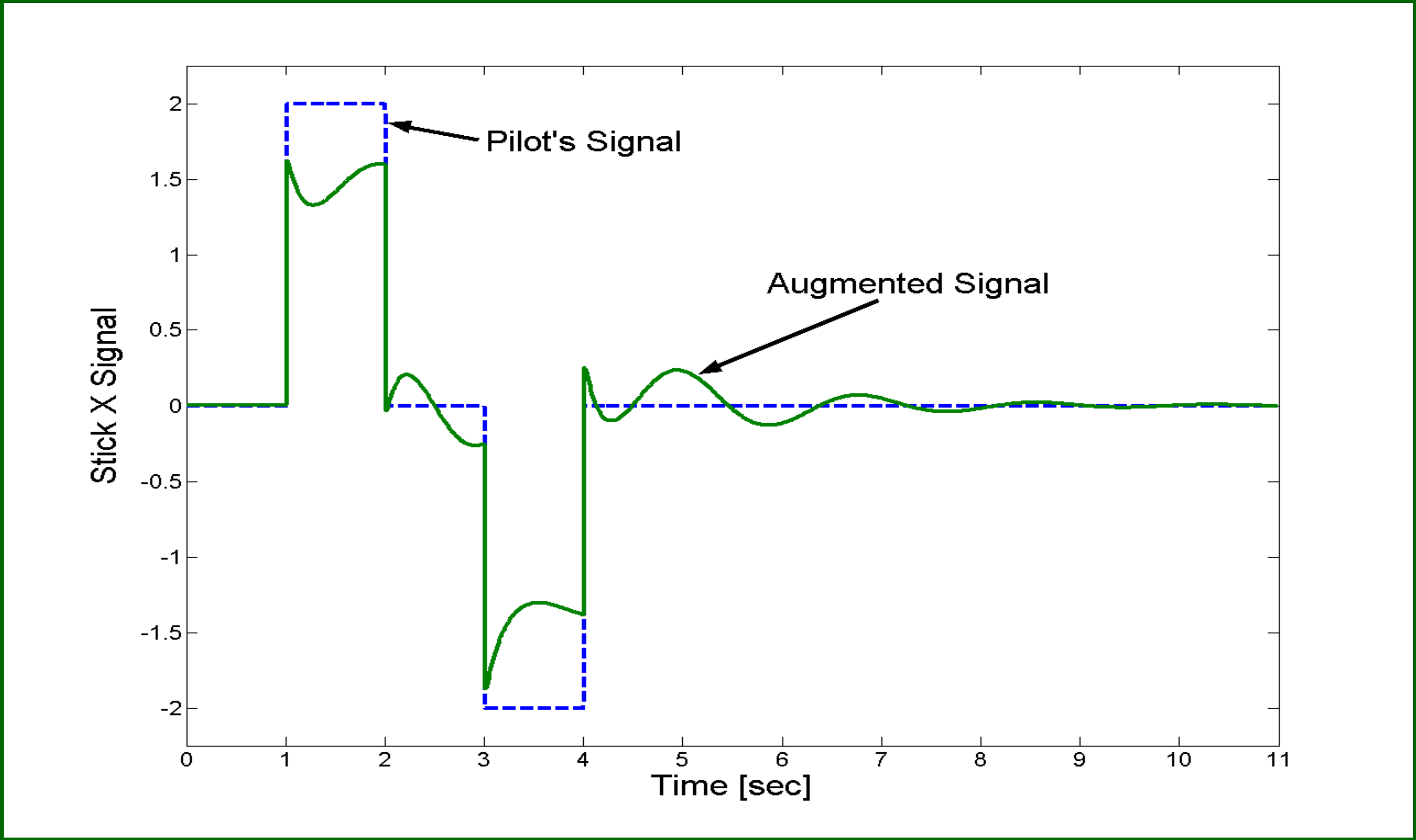


[To example →]

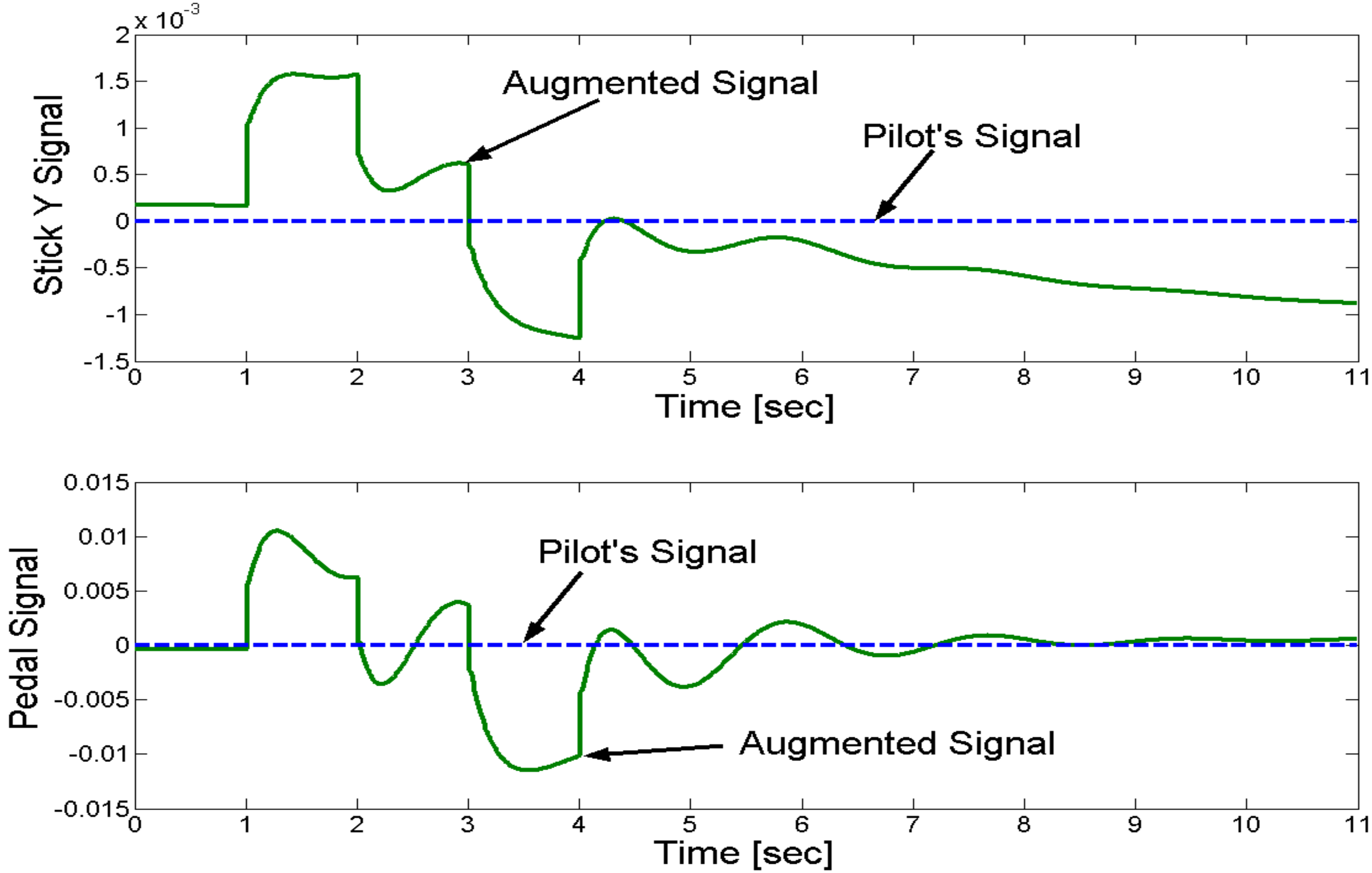
Employ ADP for Design of Optimal Controller, an Example:

Control Augmentation System for aircraft.

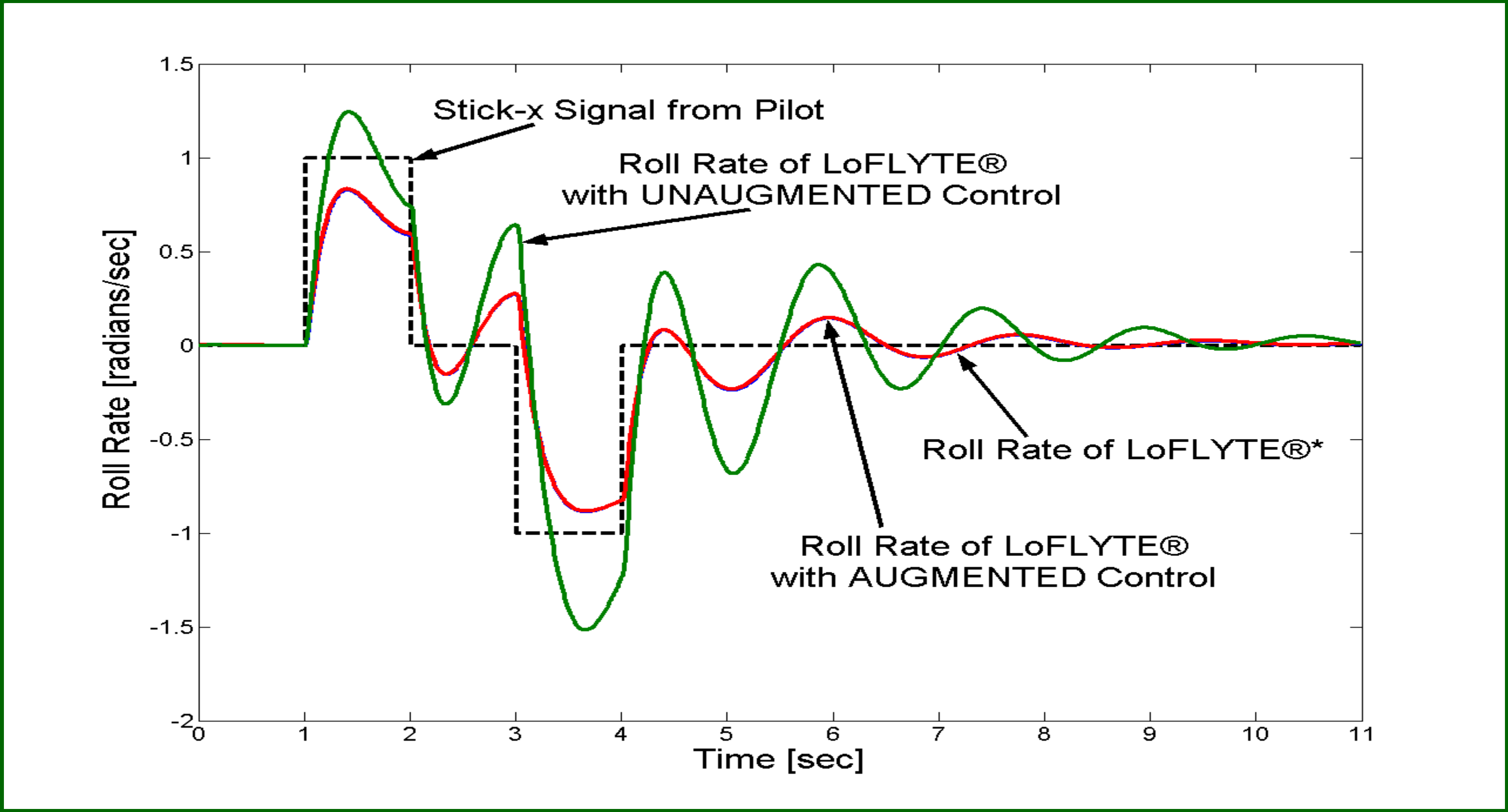




Stick-x doublet: pilot’s stick signal vs. augmented signal
(the latter is sent to aircraft actuators)



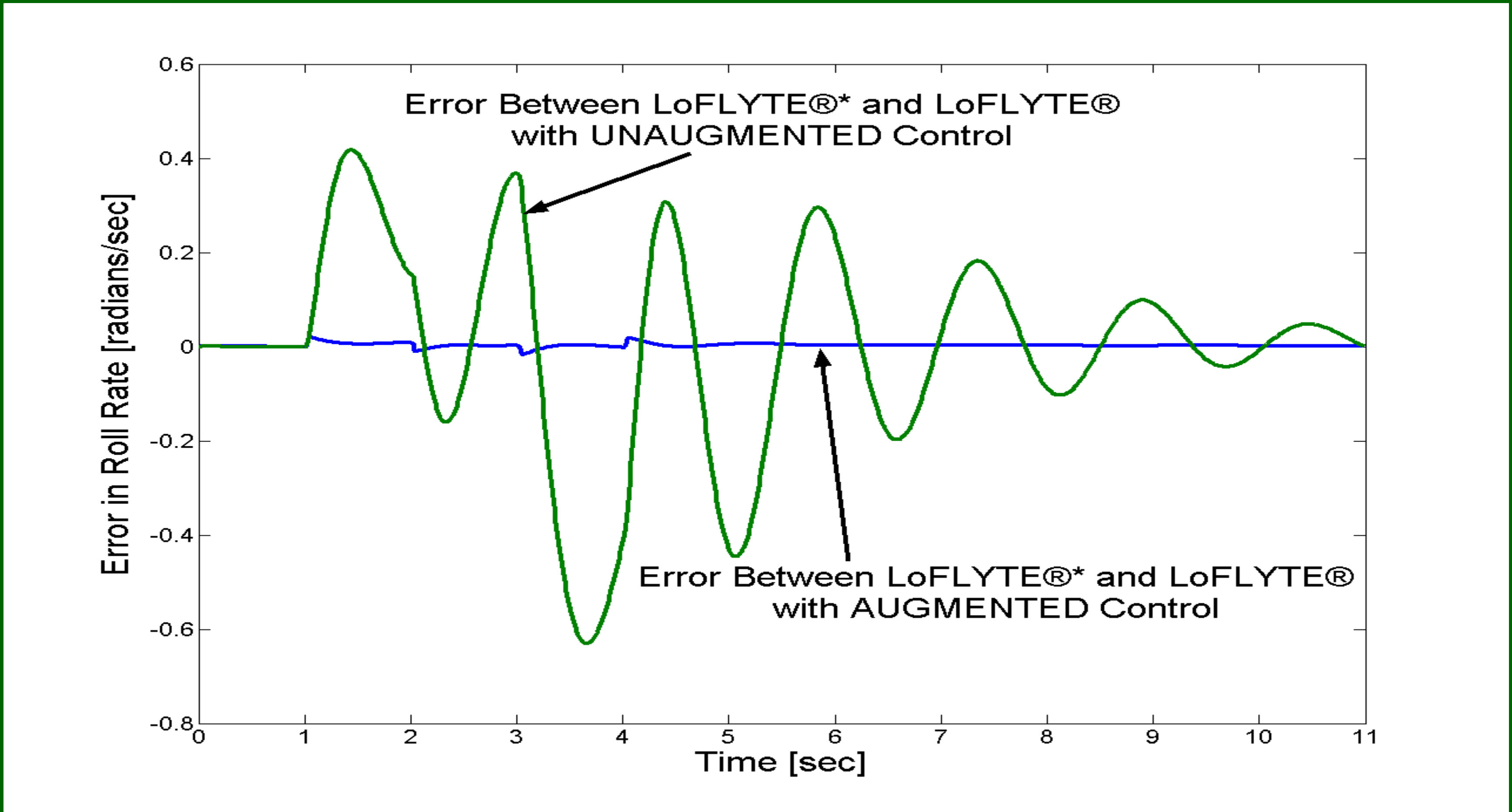
Augmentation commands for stick-y and pedal that the controller learned to provide to make the induced a) pitch (stick-y) and b) yaw (pedal) responses of LoFLYTE® **match those** of LoFLYTE®*



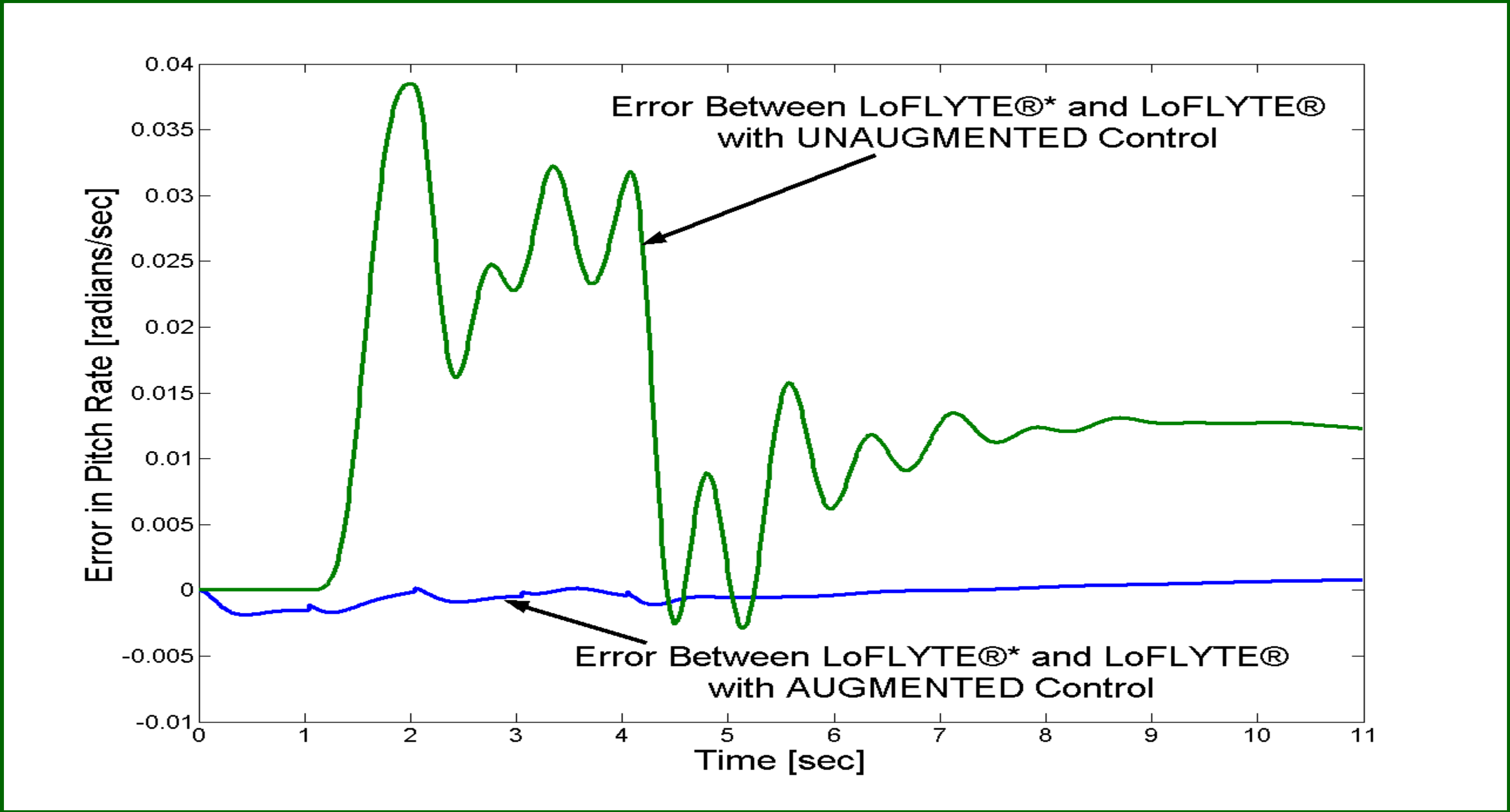
Green: Unaugmented Red: Augmented Control Blue: Reference

Pilot stick-x doublet signal (arbitrary scale in the Figure), and roll-rate responses of 3 aircraft: LoFLYTE® w/Unaugmented control, LoFLYTE® w/Augmented Control, and LoFLYTE®*.

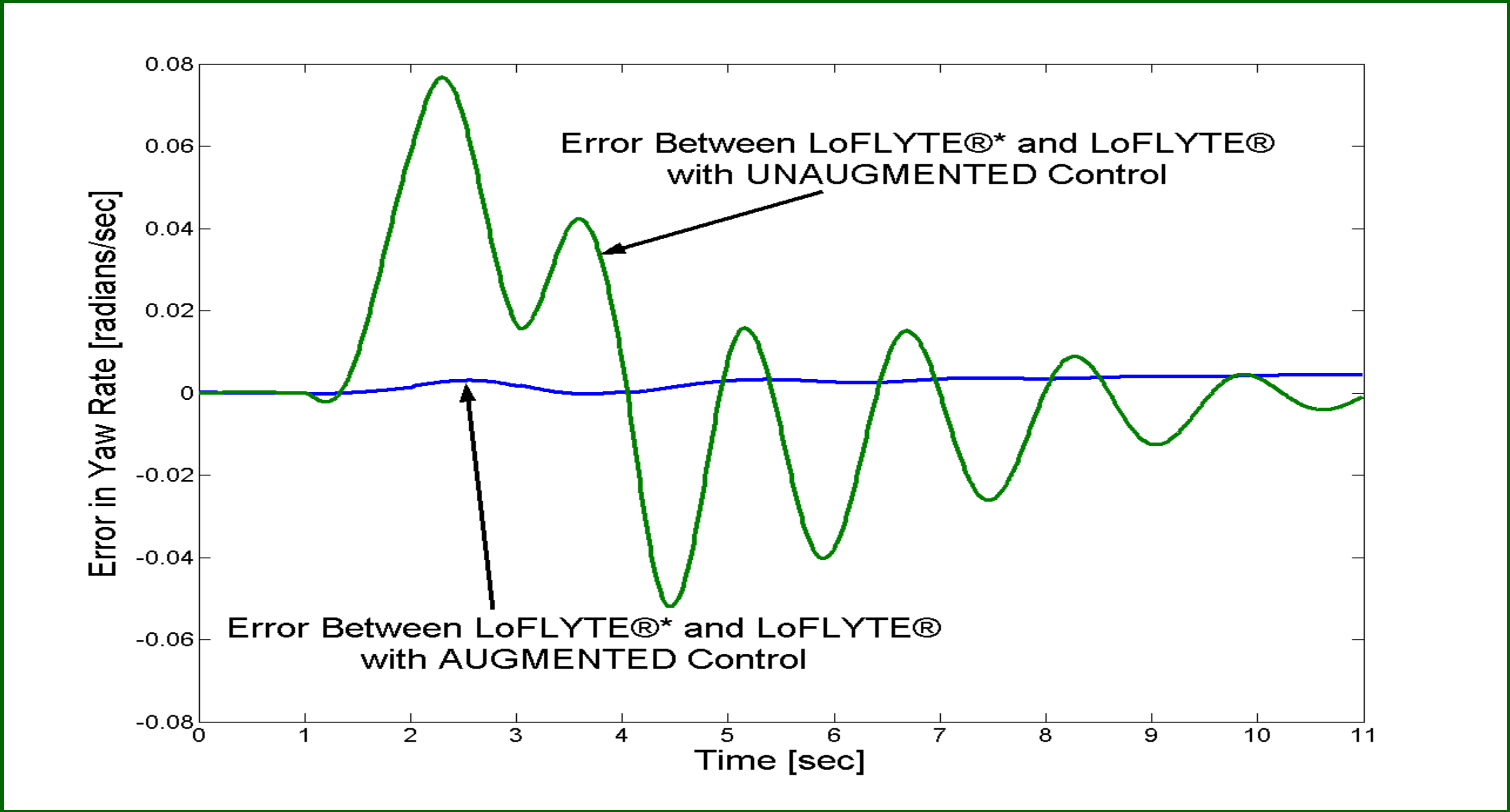
(Note: Responses of latter two essentially coincide.)



Roll-rate error (for above stick-x signal)
between LoFLYTE®* and LoFLYTE® w/Unaugmented Control, and
between LoFLYTE®* and LoFLYTE® w/Augmented Control signals



Pitch-rate error (for above stick-x signal)
between LoFLYTE®* and LoFLYTE® w/Unaugmented Control, and
between LoFLYTE®* and LoFLYTE® w/Augmented Control signals.



Yaw-rate error (for above stick-x signal)
between LoFLYTE®* and LoFLYTE® w/Unaugmented Control, and
between LoFLYTE®* and LoFLYTE® w/Augmented Control signals.

- Blue: LoFLYTE® w/ **Un**augmented control
- Red: LoFLYTE® w/Augmented Control
- Black: LoFLYTE®*

Roll 1

Order of presentation in this talk:

1. Controls (including some historical aspects)
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. Higher-Level Application of ADP (to controls)
6. System Identification
7. Examples
8. Concluding comments

Notion of “higher level”:

1. Entails augmenting our thinking about how we apply ADP in control applications.
2. We introduce into the process a meta-level observer (agent) to implement context monitoring.

CONTEXT:	*Plant*	*Environment*	*Criterion Function*

3. Applies ADP to a different optimization problem: that of *selecting* a controller from the experience repository described earlier corresponding to discerned context.

Notion of Higher Level, cont.

4. If the Agent discerns that context has changed (in one or more of its components), then it
   a. Determines what the context changed to, and
   b. Selects corresponding controller from its "experience repository".

   Agent's activities are said to occur at a "higher level" (from the one normally employed in application of ADP).

5. Entails meta-level analysis of problem domain to determine the context variables for the agent to monitor.
6. Set up agent to <u>measure</u> or <u>calculate</u> values for these context variables (CVs).

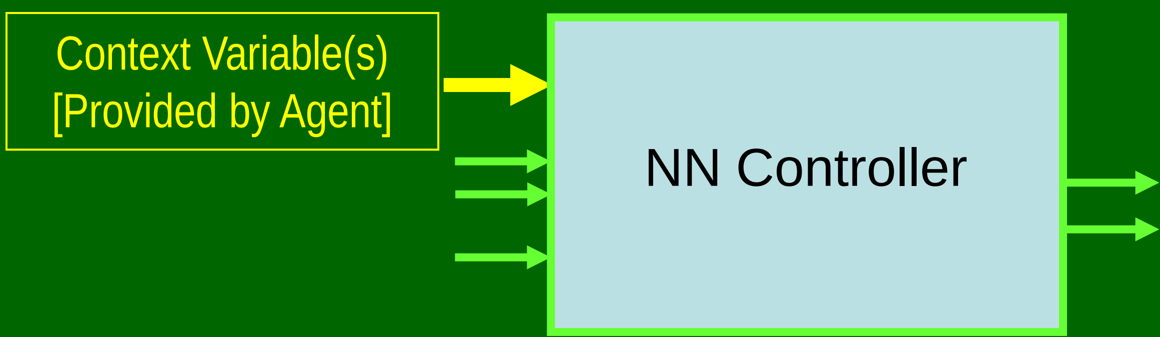
First step toward "higher level" approach:

Agent provides NN with CV values during training via ADP.

Recall the Standard Use of ADP:

NN Controller is Designed/Trained via ADP

NN Controller is Designed via ADP with auxiliary CV variables.



"Contextually Aware Controller"

[Results in multiple embedded R(t) → u(t) controllers.]

[In operation, CV serves as SELECTOR for the different Controllers.]

Notion of Higher Level, cont.

Previously developed examples of Agent providing NN with CV values during training via ADP :

1.) **Steering controller for autonomous four wheel vehicle to change lanes.**

Employ standard state variable inputs plus context variable CV = calculated estimate of current coefficient of friction between tire and road. Deals with patch of ice on road.

2.) **Control Augmentation System for aircraft.**

Employ standard state variable inputs plus context variable CV = calculated estimate of current location of center of gravity. Deals with sudden change of c.g.

[Continue the previous aircraft example:]

Notion of Higher Level, cont.

Center of gravity issue:

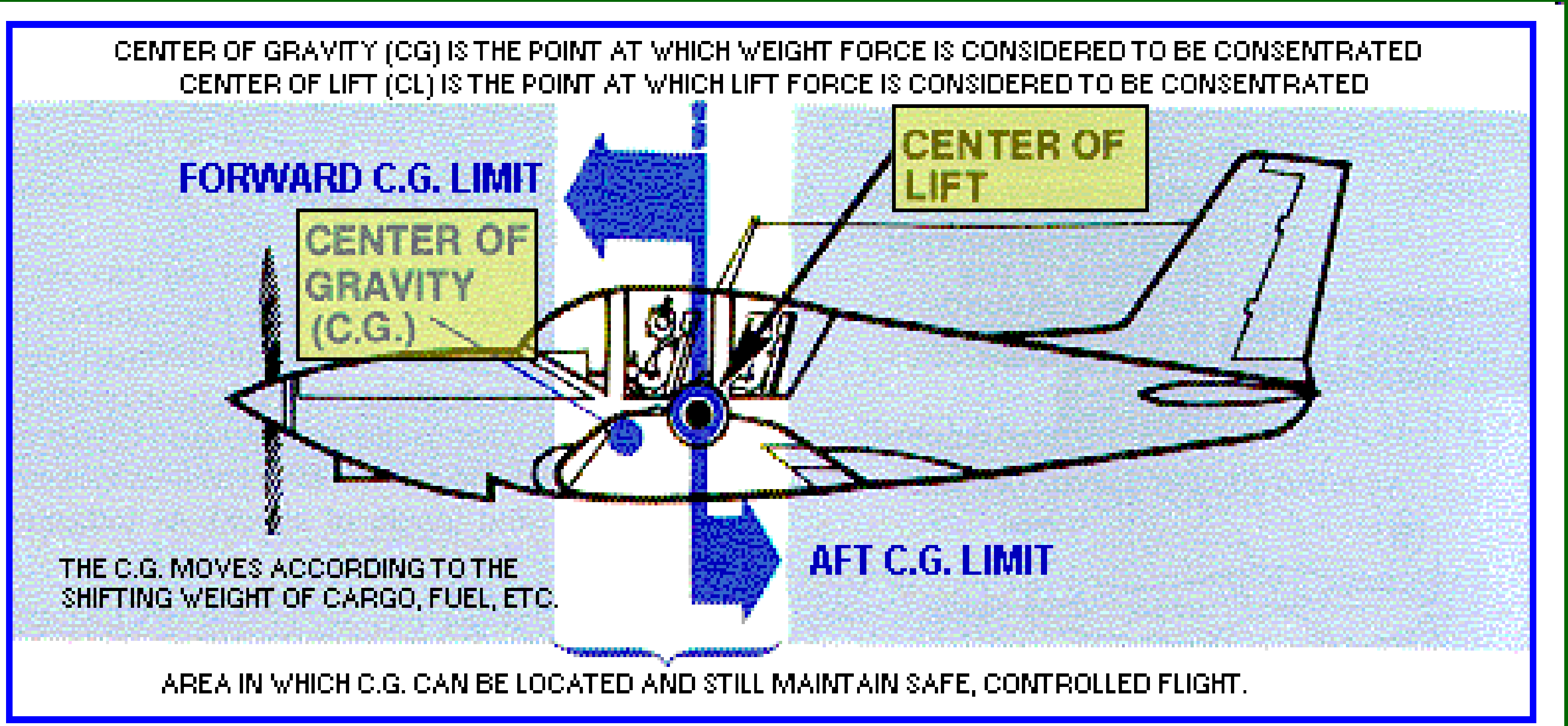


Figure 4-6 Center of gravity and center of lift.

http://www.allstar.fiu.edu/aero/flight43.htm

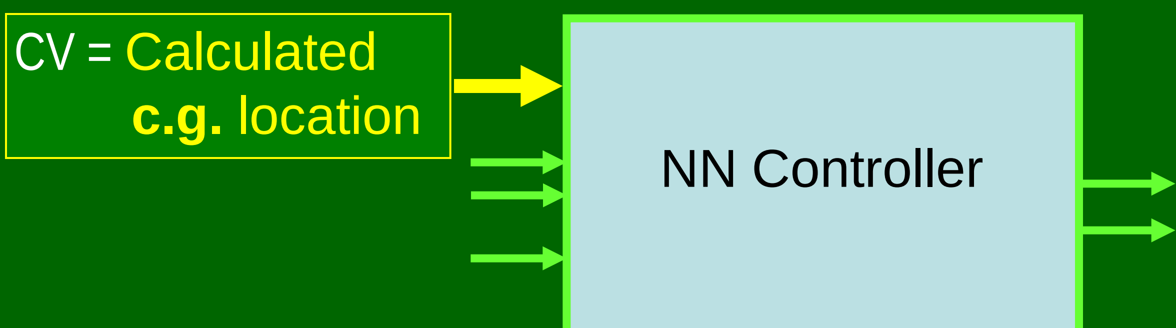
First step toward "higher level" approach:

Agent provides NN with CV values during training via ADP.

Recall the Standard Use of ADP:

NN Controller is Designed/Trained via ADP

NN Controller is Designed via ADP with auxiliary CV variables.



"Contextually Aware Controller"

[Results in multiple embedded R(t) → u(t) controllers.]

[In operation, CV serves as SELECTOR for the different Controllers.]

- Blue: LoFLYTE® w/ **Un**augmented control
- Red: LoFLYTE® w/Augmented Control
- Black: LoFLYTE®*

Pitch w/ cg Shift

NEXT step toward "higher level" approach:

At NWCIL, an expanded approach to *experience* is being addressed

- via a notion of *experience repository*, and

- via a novel concept for applying

Reinforcement Learning / Adaptive Critics

vis-à-vis the experience repository

→ *Higher-Level Learning Algorithm* (HLLA).

Higher Level Learning Algorithm

KEY IDEA of HLLA:

Re-purpose the Reinforcement Learning method (to a "higher level") such that

1) instead of using it to *design* an optimal controller for a given task (the "standard" way to use ADP)
2) An already achieved *collection* of such solutions for a variety of related contexts is provided (as an *experience repository*), and
3) HLLA creates a strategy for optimally *selecting* a solution from the repository.

→ [Note two different uses of term optimal.]

Recall item #4 in earlier list related to Notion of Higher Level:

4. If the Agent discerns that context has changed (in one or more of its components), then it
   a. Determines what the context changed to, and
   b. Selects corresponding controller from its "experience repository".

For REMAINDER OF TALK:

Assume that of three Context components, Plant is allowed to change but the Environment and CF portions remain fixed.

IMPLIED NEXT TASK:

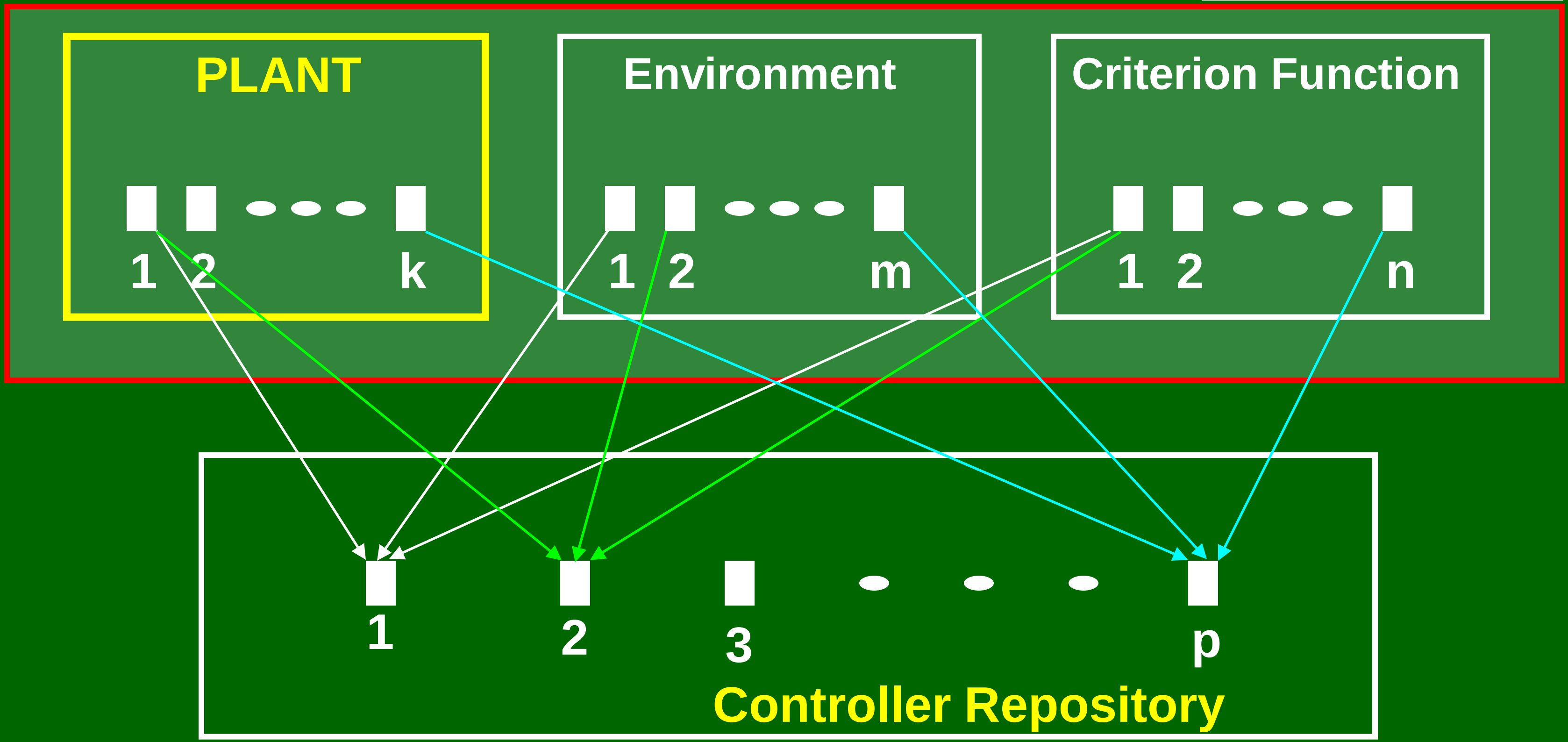
After Agent determines Context has changed, do 4a above – i.e., Perform System Identification to determine what plant has changed to.

THE HLLA APPROACH IS APPLICABLE TO THIS TASK TOO!

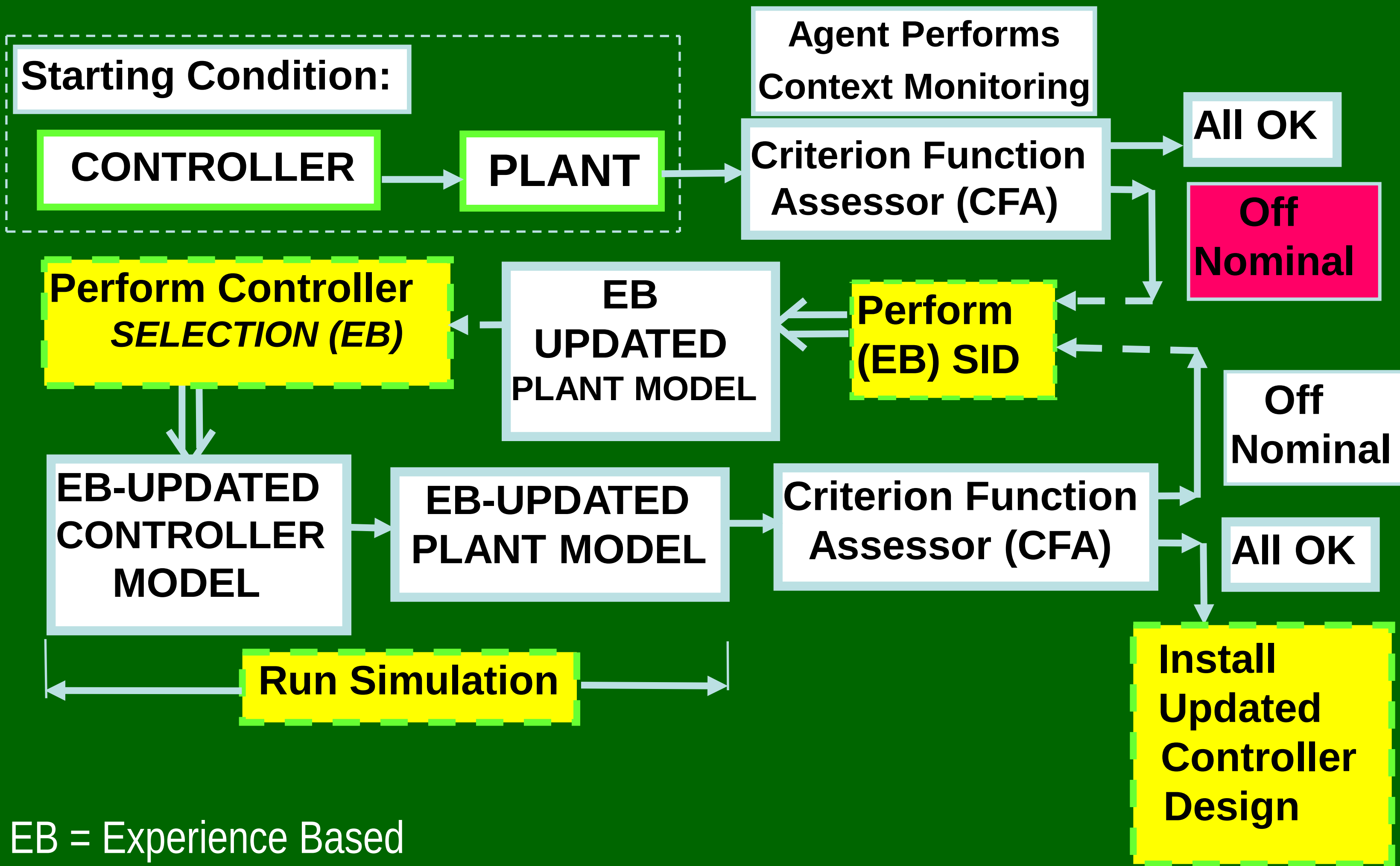
Order of presentation in this talk:

1. Controls (including some historical aspects)
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. **Higher-Level Application of ADP** (to controls)
6. **to System Identification**
7. Examples
8. Concluding comments

For next slide, recall this definition of → **Context**

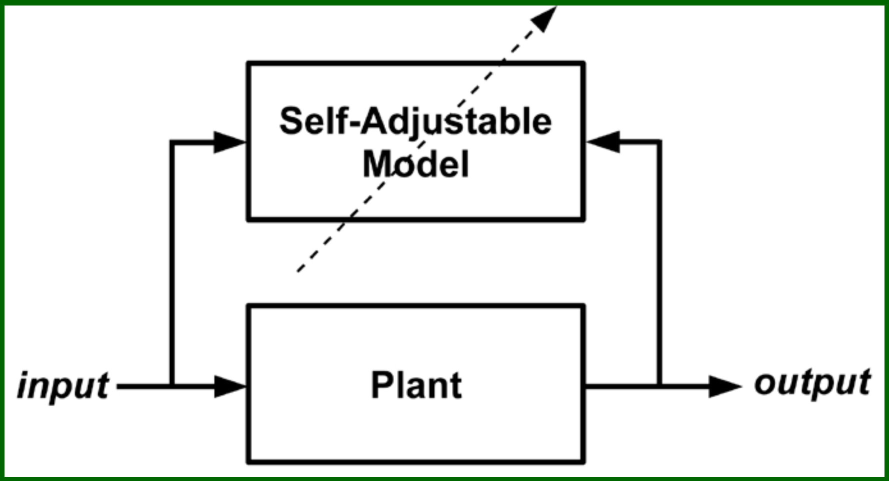


Overview of "higher level" approach for case of plant changes:



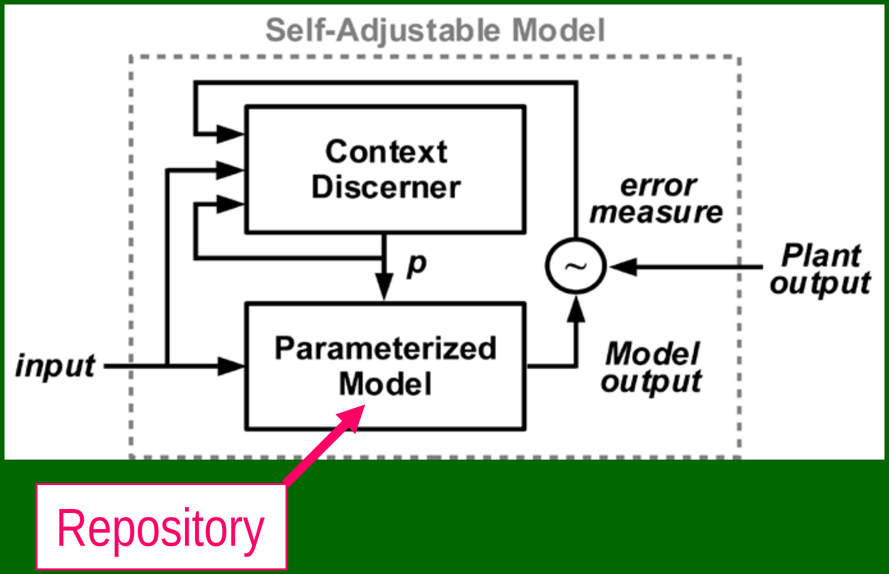
EB = Experience Based
SID = System Identification

Overview of HLLA SysID process:



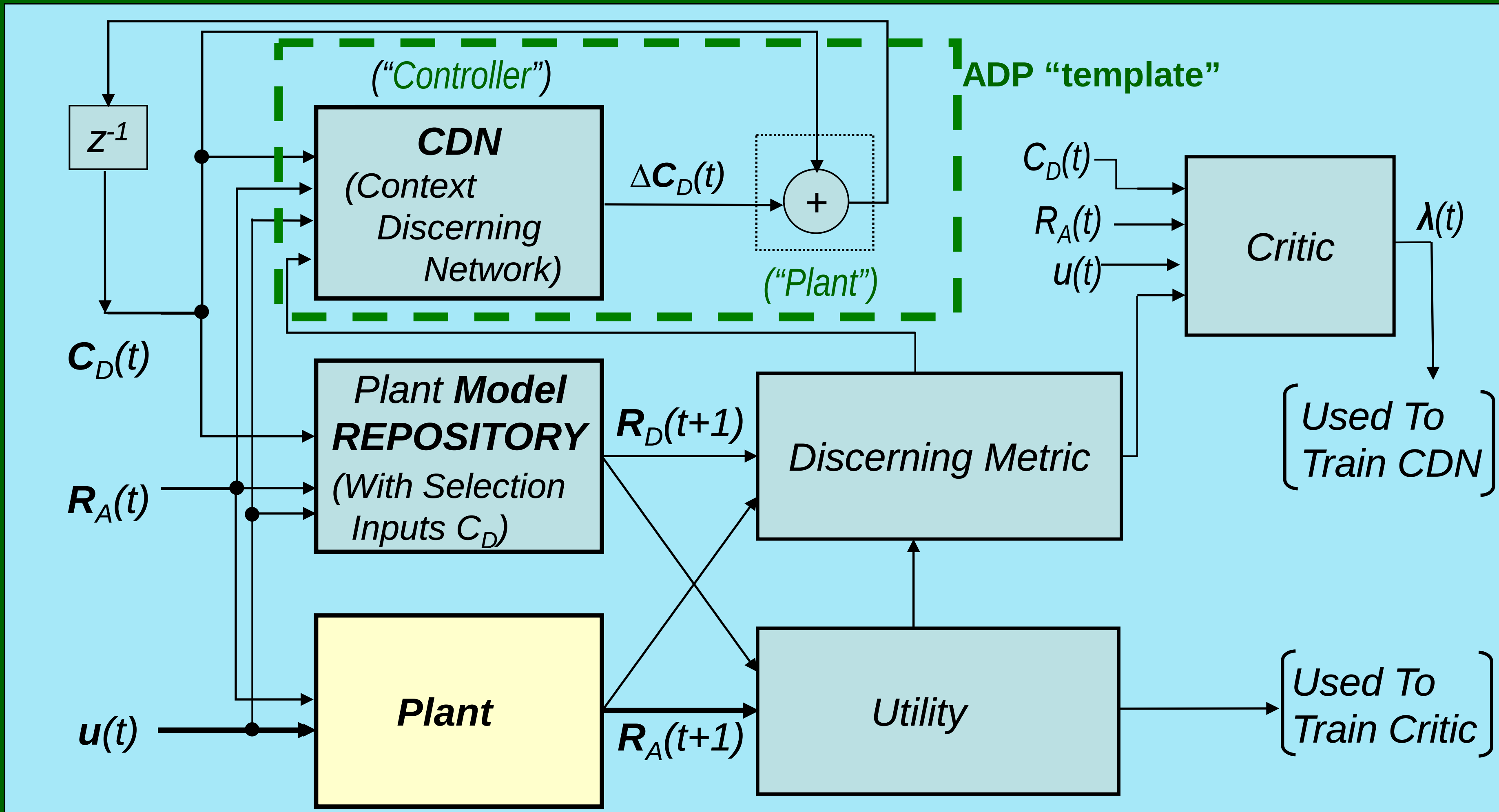
Characterize as a Self-Adjustable Model.

The Self-Adjustable Model monitors the input and output of the Plant to determine whether or not the Plant has changed and, if it has, what it has changed to.



The **context discerner** (CD) provides the parameter values *p* ('selector input') that instantiate a specific mapping in the parameterized-model box. After the CD has learned a family of mappings, it selects a specific mapping based on a measure of the difference between model's output with that of the plant being observed. The CD is trained via an Adaptive-Critic-type of Approximate Dynamic Programming approach (not shown).

Training the CDN to Discern Plant Status (SysID) Optimally:



__ADP "Plant"__: **u**(t) ⟷ Δ**C**$_D$(t)

R(t) ⟷ **C**$_D$(t)

R(t+1) = **C**$_D$(t)+Δ**C**$_D$(t)

Order of presentation in this talk:

1. Controls (including some historical aspects)
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. **Higher-Level Application of ADP** (to controls)
6. to System Identification
7. Examples
8. Concluding comments

HLLA Stage 1 (System Identification - SysID)

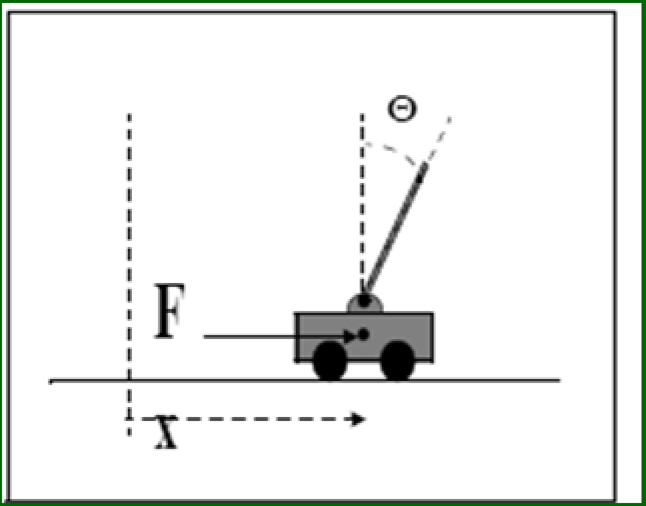
1. Experiment 1: Proof of Concept via Equation as Plant
   a) Pole Cart Problem
   b) CDN learned to discern mass and length from motion data
2. Experiment 2: Proof of Concept via NN as Plant
   a) Multiple Context Variables
   b) Demonstrated HLLA principle can work
3. Experiment 3: Refined Exploration via NN as Plant
   a) Single adjustable parameter
      i. Noise-Free & Perfect Model
      ii. Noisy Measurement Data
      iii. Imperfect Model
   b) Two adjustable parameters
      i. Noise-Free & Perfect Model
      ii. Noisy Measurement Data

HLLA Stage 1 (System Identification - SysID)

Experiment 1: Proof of Concept via Equation as Plant

Assume:

1) A controller for nominal Pole-Cart is in operation.
2) Sudden change of pole mass and length.



3) For controller to “adapt”, needs to find present condition of the Pole-Cart.
4) CDN discerns mass and length of the pole directly from motion data.

HLLA Stage 1 (System Identification - SysID)

Experiment 1: Proof of Concept via Equation as Plant

Method:

1) Craft a "repository" of various versions of the Pole-Cart plant.
2) Develop HLLA process to optimally select (with respect to efficiency and effectiveness of selection process) a model from the repository that matches current plant condition.

HLLA Stage 1 (System Identification - SysID)

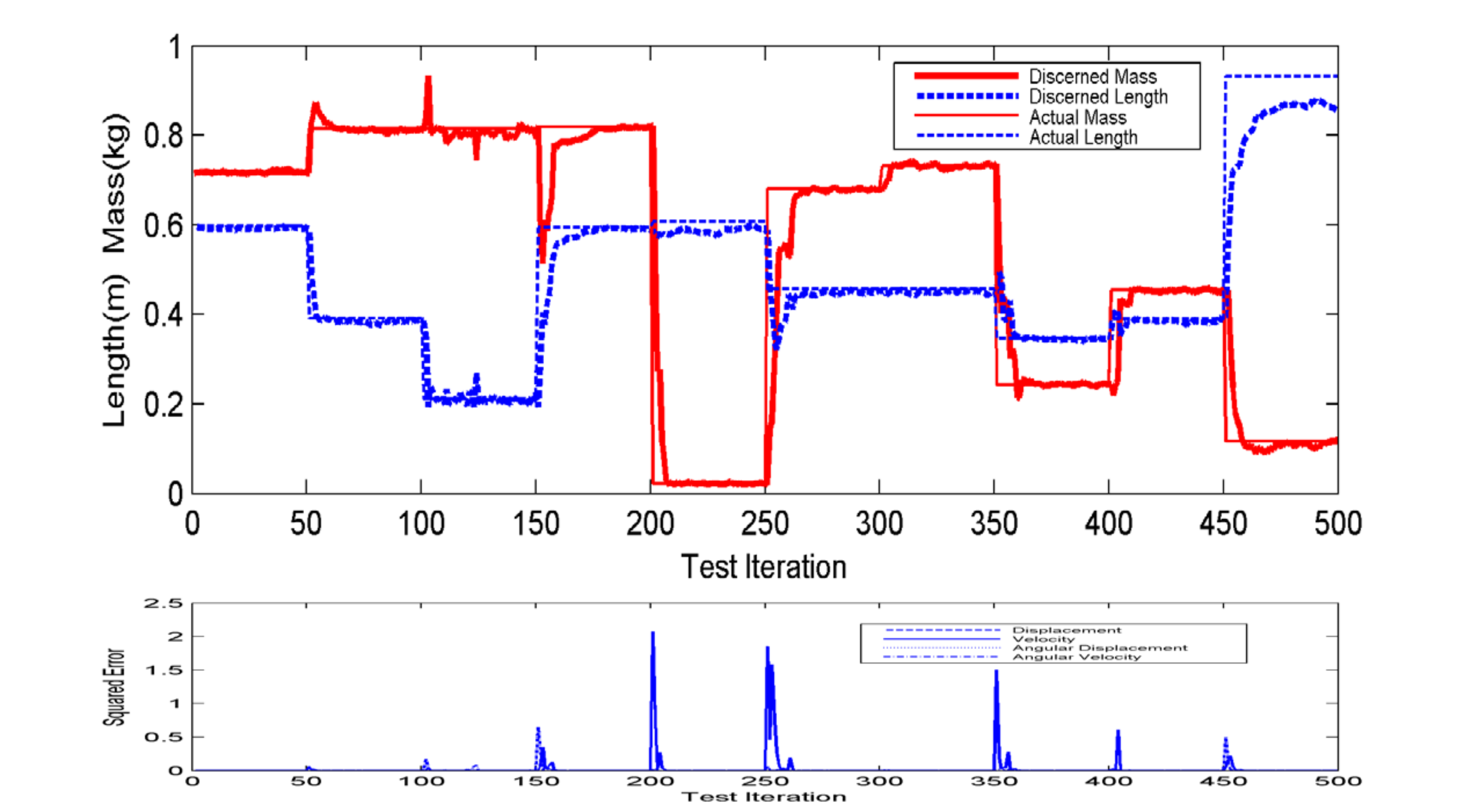
Experiment 1: Proof of Concept via Equation as Plant

Approach Taken

- Employed equations of motion of Pole-Cart plant to populate the "repository".
- Changes in plant are accomplished via changes in parameter values of the equations.
- Only mass and length parameters are employed to index the plant models in the repository (for present experiments).

HLLA Stage 1 (System Identification - SysID)

Experiment 1: Proof of Concept via Equation as Plant



TOP: Context Discernment in response to context change (change in plant par. values) every 50th iteration.
BOTTOM: Errors between pole-cart system state variable and models selected during discernment process.

HLLA Stage 1 (System Identification - SysID)

Experiment 2: Proof of Concept via NN as Plant

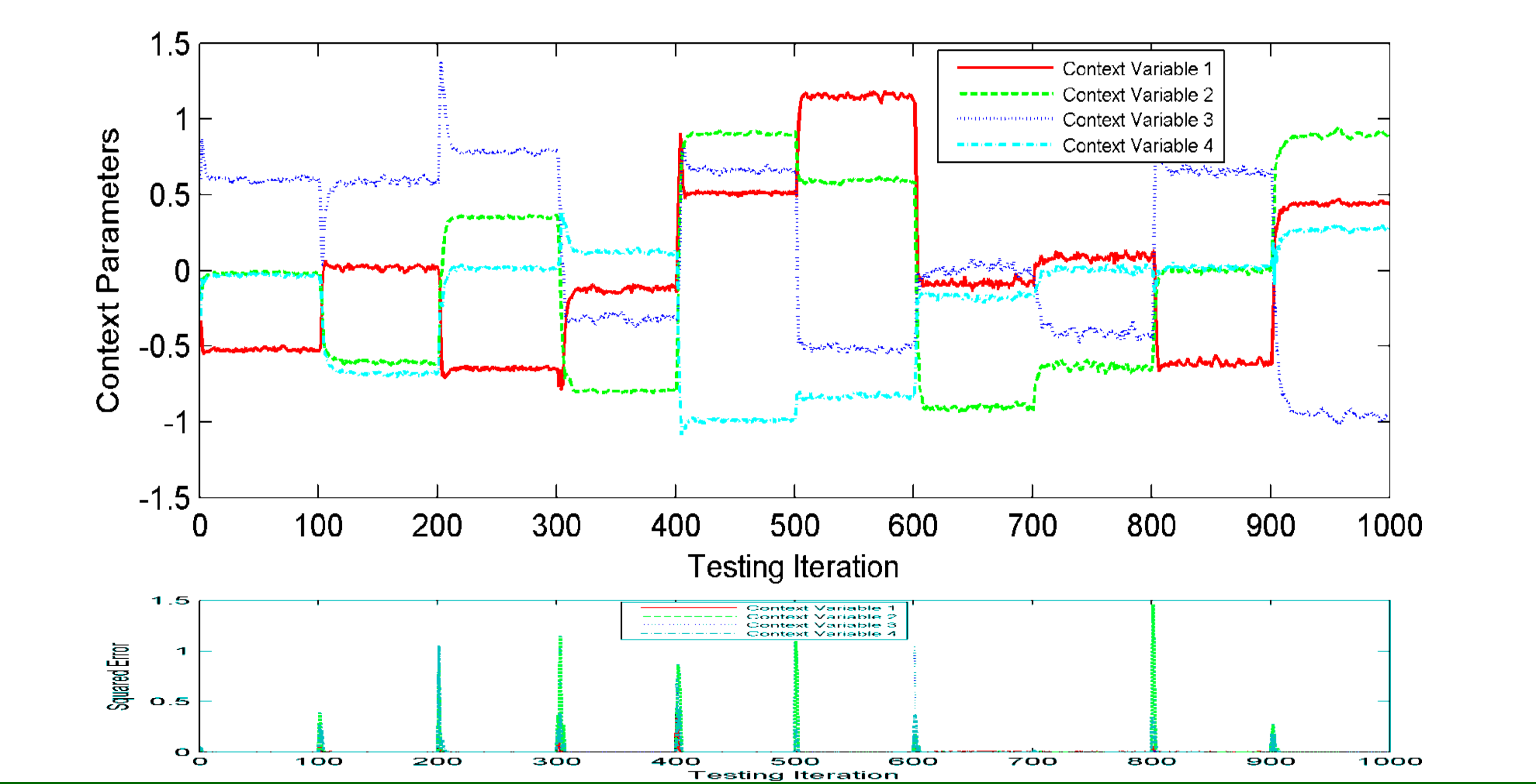
Approach taken:

- Crafted a neural network of specified structure and element type to populate the “repository”.
- Changes in plant accomplished via changes in selected weight values of NN.
- Weights of NN are here considered “parameters” of the plant.

[Overall HLLA process is same as described previously.]

HLLA Stage 1 (System Identification - SysID)

Experiment 2: Proof of Concept via NN as Plant



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

BOTTOM: Errors between pole-cart system state variable and models selected during discernment process.

HLLA Stage 1 (System Identification - SysID)

Experiment 3: Refined Exploration via NN as Plant

Explore effects on process of training CDN and performance of CDN under conditions of:

1) Single Adjustable Parameter
    a) Noise-Free & Perfect Model of Plant
    b) Noisy Measurement Data
    c) Imperfect Plant Model
2) Two Adjustable Parameters
    a) Noise-Free & Perfect Model
    b) Noisy Measurement Data

RESULTS SUBMITTED TO IJCNN-2011

HLLA Stage 1 (System Identification - SysID), cont.

I show just one slide from those results, because they provide a nice demonstration of the CDN’s accomplishment.

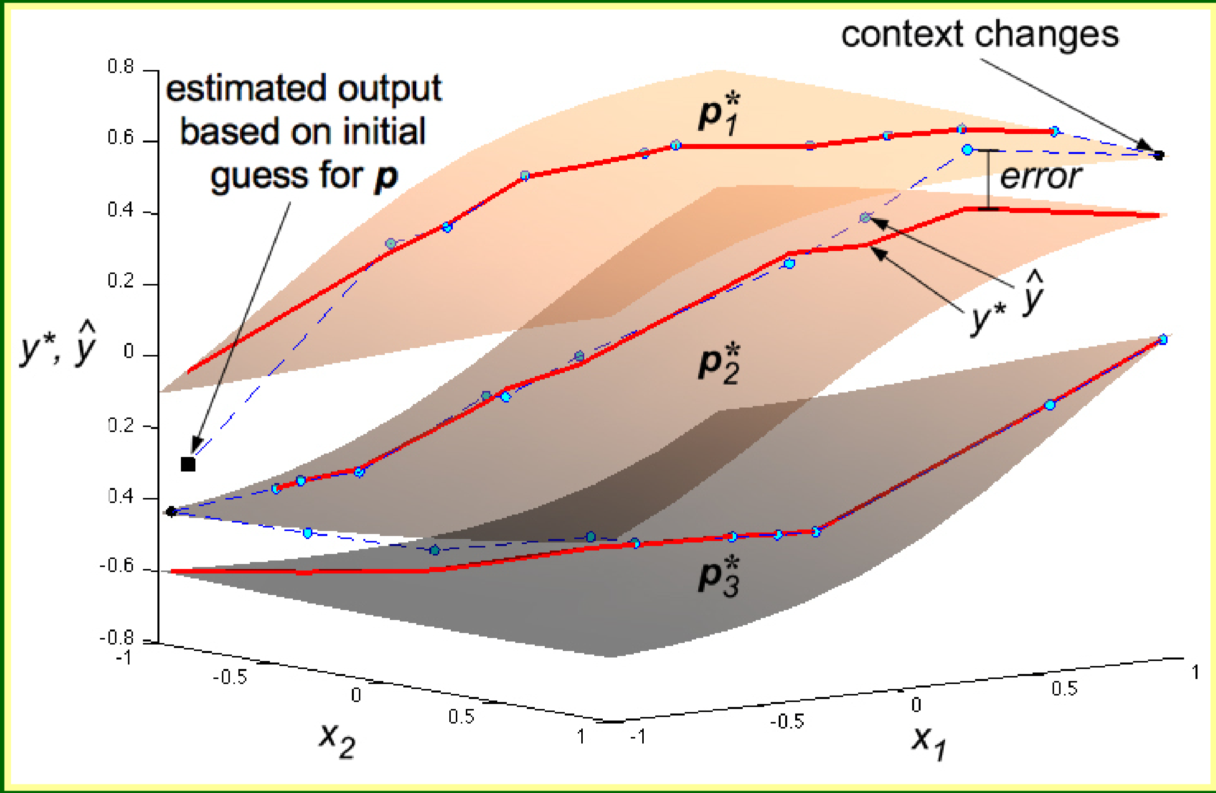
The “NN as plant” test bed allows a nice representation of the operation of the CDN:

The set of fixed weights and structure of the NN implement a family of mappings (surfaces); the NN’s variable weights serve to “index” the different surfaces.

Under guidance of the ADP process, the CDN learned to index and optimally select the appropriate mapping based on a (relatively) small observation window.

HLLA Stage 1 (System Identification - SysID)

Experiment 3: Part 1: NN with Single Adjustable Parameter



Noise-Free, Perfect Model: The three indicated surfaces correspond to three selected bias values (parameters *p**) for a family of mappings with a *particular* instantiation of the fixed weights.

HLLA Stage 1 (System Identification - SysID)

General Results: HLLA Stage 1 (SysID):rea

- Results indicate that the HLLA approach can be robust and adaptive when performing system identification tasks.
- Demonstrations so far have been on plants represented by low-order differential equations and/or on small neural networks.
- Latest experiments include addition of measurement noise, and (slightly) imperfect models.
- Agents using this approach have achieved:
  - a) high levels of performance, even with rather large amounts of noise, and
  - b) reasonable performance when employing imperfect models.

HLLA Stage 1 (System Identification - SysID)

Four insights gained from these experiments:

1. training process adopted can significantly affect subsequent performance;
2. characteristics of the plant/system to be identified affects the CD's ability to identify it;
3. performance may still be satisfactory for even large amounts of noise; and
4. performance may be satisfactory with an imperfect model.

* These all correspond well with our intuition about human learning.

Order of presentation in this talk:

1. Controls (including some historical aspects)
2. Adaptive Critic type of Reinforcement Learning
3. Dynamic Programming
4. Adaptive Dynamic Programming
5. **Higher-Level Application of ADP** (to controls)
6. to System Identification
7. Examples
8. Concluding comments

Concluding Comments:

- What about the question implied in title of paper:

  Might HLLA be a basis for a new phase in evolution of the controls field?

- The Controls Field has a rich history – through various phases each associated with identifiable tools, ideas, ways of thinking.
- I suggest HLLA is a new way of thinking about application of the ADP methods.
- So, ????

Concluding Comments, cont.

I am phasing down my academic career and entering a new era of my life after this school year.

I firmly believe there are tremendous possibilities for this line of research, and I urge those of you early or mid career to consider entering it.

Key ideas:

- EXPERIENCE (as memory of solutions)
- Notion of CONTEXT, with three components
- Context Discernment via meta-level agent
- Maintain explicit memory of previous solutions for variety of context instantiations (in a searchable repository)

Concluding Comments, cont.:

- HLLA is a “point of view” – on part of researcher/developer/implementer.
- Optimization problem turns into one of how to best *select* controller from experience repository.
- “Think higher”, in sense of crafting the optimization task in a way performable by ADP methods.
- Study the human exemplar for hints on “human-like” control.
- HLLA method is applicable to the SysID problem too.

I suspect the mathematics of geometric topology will turn out being useful in this research (manifolds, etc.).

While the above comments focus on the HLLA approach to designing selecting strategies, I believe the “Contextually Aware Controller” approach also has substantial promise.

Questions?